(2)

AD-A255 203

Research and Development Technical Report
SLCET-TR-92-8

Numerical Analysis of Permittivity With Loss in Isotropic Binary Composites

Stephen R. Wallin
Marta J. Wallin
University of Southern Colorado

and

John Kosinski
Arthur Ballato
Electronics Technology and Devices Laboratory

June 1992

DISTRIBUTION STATEMENT

Approved for public release.
Distribution is unlimited.

U.S. ARMY LABORATORY COMMAND
Electronics Technology and Devices Laboratory
Fort Monmouth, NJ 07703-5601

NOTICES

Disclaimers

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
June 1992

3. REPORT TYPE AND DATES COVERED
Technical Report: May-Aug 1991

4. TITLE AND SUBTITLE
NUMERICAL ANALYSIS OF PERMITTIVITY WITH LOSS IN ISOTROPIC BINARY COMPOSITES

5. FUNDING NUMBERS
C: DAAL03-86-D-0001
Delivery Order 2355

6. AUTHOR(S)
Stephen R. Wallin and Marta J. Wallin (Univ. of Southern Colorado); John Kosinski and Arthur Ballato (ETDL)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Laboratory Command (LABCOM)
Electronics Technology and Devices Laboratory (ETDL)
ATTN: SLCET-MA
Fort Monmouth, NJ 07703-5601

8. PERFORMING ORGANIZATION REPORT NUMBER
SLCET-TR-92-8

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The use of composite materials and structures to provide characteristics unattainable directly from the constituent materials is well known. Work has been undertaken to apply this principle to the development of new dielectric materials for use in capacitors with greater energy density, lower loss, and higher breakdown resistance. This report describes the numerical analysis of permittivity in isotropic binary composite dielectrics through three-dimensional computer simulation.

14. SUBJECT TERMS
Numerical analysis; composite materials; dielectric materials

15. NUMBER OF PAGES
125

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
UL

CONTENTS

FIGURES

FIGURES (cont.)

FIGURES (cont.)

INTRODUCTION

The use of composite materials and structures to provide characteristics unattainable directly from the constituent materials is well known. Perhaps the most widespread example is steel-reinforced concrete wherein the high tensile strength of steel in conjunction with the high compressive strength of concrete yields a composite material with structural properties far superior to those of either constituent. More recently, work has been undertaken to apply this principle to the development of new dielectric materials for use in capacitors with greater energy density, lower loss, and higher breakdown resistance [1]. High permittivity dielectrics are necessary to achieve high energy densities, however high permittivity is usually associated with high loss. Low loss is similarly associated with low permittivity and low energy density dependent on loss tangent. Composite dielectrics of appropriate combinations of constituent materials may show high overall permittivity with small loss even though one or more of the constituents has large individual loss. This is known as the Maxwell-Wagner effect [2-4].

The net dielectric behavior of a randomly interspersed composite is dependent on the spatial dimensionality (1-D vs. 2-D vs. 3-D), domain geometries (domain size, domain shape, stratification, etc.), interconnection effects (percolation), and fractalization (interfaces or connectedness per unit volume). A self-similarity averaging law which is useful to the dielectric engineer is Lichtenecker's formula [5,6]

$$\epsilon^{\alpha} = \sum_k \epsilon_k^{\alpha} v_k \tag{1}$$

where ϵ is the resultant permittivity of the composite, α is an exponential averaging factor ($-1 < \alpha < +1$), and the summation is over the constituent species with permittivity ϵ_k and volume fraction v_k respectively. This formula can be justified theoretically to apply to random (self-similar, scaleless) and history invariant composites. The permittivity ϵ is defined fully in terms of the electric and displacement fields and is frequency dependent. The factor α is referred to as the exponential averaging factor or self-similarity factor. The behavior of this factor has certain values in special limiting cases and was originally called a "formzahl" (form number) by Wiener [7]. In the case of isotropic flat layers with surface normals oriented parallel to the applied electric field, $\alpha = -1$, and for isotropic flat layers with surface normals perpendicular to the applied electric field direction, $\alpha = +1$. The mathematical interpretations of the exponential averaging factor for the following values are: +1 = 'arithmetic' averaging, 0 = 'geometric' (or logarithmic) averaging, and -1 = 'harmonic' averaging. In regard to the physical problem of randomly interspersed dielectric composites with a self-similarity, the values of the exponential averaging factor are only known to lie within these bounds [7-12]. The exponential averaging factor α however does offer a useful way of presenting results and general

trends in a comparative sense to one another.

A second factor which is also bounded is the depolarization factor 'A'. The depolarization factor is a geometrical factor that arises from a self-consistent treatment of the dielectric problem where ellipsoidal domains are each surrounded by an effective medium host. This picture is usually the lowest order scattering approximation where the dielectric grains are much smaller than any associated electromagnetic wavelengths. The actual composite grains may be distant from the ellipsoidal shape, especially when dealing with mixtures not dominated strongly by any particular species. This approach is referred to as the effective-medium-theory coherent-potential-approximation (EMT-CPA) and has been derived in several manners [13-15]. It is most succinctly expressed as

$$\sum_k \frac{\epsilon_k - \epsilon}{A\epsilon_k - (1-A)\epsilon} v_k = 0 \ . \tag{2}$$

The depolarization factor 'A' can be determined by an integral which is taken over the shape of the ellipsoid and is discussed in Appendix A [11, 16-18]. In the case of a spheroid situated in N spatial dimensions the depolarization is simply $A = 1/N$.

Our analysis will be presented in both forms: the exponential averaging factor 'α' and the depolarization factor 'A'. The exponential averaging factor 'α' may be thought of as a measure of the degree between series and parallel-like extrema or Wiener bounds of the composite. The depolarization 'A' is a measure of geometric grain shape in terms of the effective-medium-theory coherent-potential-approximation (EMT-CPA). In the limit where the constituents of the composite are dielectrically infinitesimally close, the relation $\alpha \approx (1 - 2A)$ holds. It is interesting to note that equations (1) and (2) are first order approximations to each other in the close constituent permittivity limit even though the formulae appear quite different. However, when constituents having largely different permittivities are examined using our numerical program, the composite mixture formulae yield different predictions [1]. Both methods allow us to present our numerical simulation results in a comparative way which is useful for both theoretical and experimental analysis.

ELECTRIC DISPLACEMENT FIELD

The electric field $\mathbf{E}(\mathbf{r},t)$ is defined as the electric force per unit charge acting on a stationary test charge located at point $\mathbf{r}$ and at time t (Boldface notation will be employed throughout this report to represent vector and tensor quantities). The displacement field $\mathbf{D}(\mathbf{r},t)$ as used in this report is the universal displacement field which arises from the combination of Gauss's Law with charge continuity (conservation). The result is a continuous flux quantity. In the standard

international (SI) MKSA unit format, these two relations are respectively

$$\nabla\cdot\mathbf{E}(\mathbf{r},t) = \rho_Q(\mathbf{r},t)/\epsilon_o \tag{3}$$

where $\nabla\cdot$ is the spatial divergence operator, $\rho_Q(\mathbf{r},t)$ is the volume charge density and ϵ_o is the permittivity of free space, and

$$\nabla\cdot\mathbf{J}_Q(\mathbf{r},t) = -\, d\rho_Q(\mathbf{r},t)/dt \tag{4}$$

where $\mathbf{J}_Q(\mathbf{r},t)$ is the charge current density (charge current per unit area). A generalized charge polarization field $\mathbf{P}_Q(\mathbf{r},t)$ is linked to the charge density by the following defining relation for $\mathbf{P}_Q(\mathbf{r},t)$

$$\nabla\cdot\mathbf{P}_Q(\mathbf{r},t) \equiv -\,\rho_Q(\mathbf{r},t) \ . \tag{5}$$

Upon partial time differentiation of equation (5) and comparing to the continuity equation (4) one finds the identity

$$\mathbf{P}_Q(\mathbf{r},t)/\ \ t = \mathbf{J}_Q(\mathbf{r},t) \ . \tag{6}$$

Segregation of charge types may be introduced when desired, but is not necessary for the derivation and application of a generalized permittivity [19] as in this report. A polarization field for discrete point charges in an inertial coordinate system from equation (5) could be

$$\mathbf{P}_Q(\mathbf{r},t) = \sum_i q_i\mathbf{r}_i \ . \tag{7}$$

In equation (7) the summation is over all charges q_i at locations $\mathbf{r}_i$ within the medium. Combining equation (3) and equation (5) yields

$$\{\epsilon_o\nabla\cdot\mathbf{E}(\mathbf{r},t) + \nabla\cdot\mathbf{P}_Q(\mathbf{r},t)\} = 0 \ . \tag{8}$$

Rearranging the brackets of equation (8) results in the divergence relation

$$\nabla\cdot\{\epsilon_o\mathbf{E}(\mathbf{r},t) + \mathbf{P}_Q(\mathbf{r},t)\} = 0 \tag{9}$$

or

$$\nabla\cdot\mathbf{D}(\mathbf{r},t) = 0 \tag{10}$$

where the quantity in the brackets becomes the definition of the generalized or universal electric displacement field [20]

$$\mathbf{D}(\mathbf{r},t) \equiv \epsilon_o\mathbf{E}(\mathbf{r},t) + \mathbf{P}_Q(\mathbf{r},t) \ . \tag{11}$$

For brevity, this generalized field will be referred to as the 'D-field' elsewhere in this report. The time derivative of the D-field gives the generalized or universal displacement current density

$$\mathbf{D}(\mathbf{r},t)/\quad t = \mathbf{J}_D(\mathbf{r},t) \ , \tag{12}$$

referred to by Maxwell as the 'true current' and the D-field can be referred to as 'true' also [21]. From the combination of fundamental laws (3) and (4) with (11) and (12), it follows that the universal current density is always divergenceless and continuous with

$$\nabla \cdot \mathbf{J}_D(\mathbf{r},t) = 0 \ . \tag{13}$$

Conversely, one might say that according to equation (12), the antiderivative (in time) of the universal current density is a vector field called the universal displacement field. The physical units of the displacement field are charge polarization per unit volume or that of surface charge density.

Dielectric measurements and analysis are often made in the frequency domain [22, 23]. In the case where the displacement field is periodic with time it can be Fourier decomposed as

$$\mathbf{D}(\mathbf{r},t) = \int \mathbf{D}(\mathbf{r},f)\ e^{+j2\pi ft}\ df \tag{14}$$

or conversely

$$\mathbf{D}(\mathbf{r},f) = \int \mathbf{D}(\mathbf{r},t)\ e^{-j2\pi ft}\ 2\pi\ dt \tag{15}$$

where $\mathbf{D}(\mathbf{r},f)$ is the electric displacement field Fourier component at frequency f, $j \equiv \sqrt{-1}$, and integrations respectively are over all f and t. The frequency domain transformation of equation (10) gives

$$\nabla \cdot \mathbf{D}(\mathbf{r},f) = 0 \ , \tag{16}$$

or equivalently from combination of equations (12) and (13)

$$\nabla \cdot \{j\ 2\pi f\ \mathbf{D}(\mathbf{r},f)\} = 0 \tag{17}$$

for $f \neq 0$ (non-static fields). The relation expressed by equation (12) becomes in the frequency domain

$$j\ 2\pi f\ \mathbf{D}(\mathbf{r},f) = \mathbf{J}_D(\mathbf{r},f) \tag{18}$$

and so it is easy to transpose between equations (16) and (17).

In most natural media, the universal electric displacement field evolves by a collective decay from the past history of electric fields which are or have been impressed upon the medium. This is expressed by the convolution

$$\mathbf{D}(\mathbf{r},t) = \int_{-\infty}^{t} \mathbf{f}(\mathbf{r},t-t') \cdot \mathbf{E}(\mathbf{r},t')\ dt' \tag{19}$$

or alternately upon change of integration variable to u defined as $u \equiv t-t'$,

$$\mathbf{D}(\mathbf{r},t) = \int_0^\infty \dot{f}(\mathbf{r},u) \cdot \mathbf{E}(\mathbf{r},t-u)\, du \tag{20}$$

where $\mathbf{D}(\mathbf{r},t)$ is the displacement field at the present time, $\mathbf{E}(\mathbf{r},t)$ is the electric field from the past to the present, the function $\dot{f}(\mathbf{r},u) = df(\mathbf{r},u)/du$ represents the history or decay correlation between the fields and $u \equiv t-t'$ is the present to past time connection variable. The function $\dot{f}(\mathbf{r},u)$ is referred to as the normalized displacement decay current or, in short, decay current. The accumulation of normalized displacement decay current is called the normalized displacement decay function $f(\mathbf{r},u)$ or, in short, decay function. The dimensions of the decay function $f(\mathbf{r},u)$ are displacement field per electric field. Causality requires that the decay current $\dot{f}(\mathbf{r},t-t')$ be zero when $t'>t$ or $u<0$ (i.e., no correlation with the future). In the time domain both the displacement and electric fields must be real valued and hence the normalized displacement decay current $\dot{f}(\mathbf{r},u)$ must also be real valued. If the medium is linear then the decay current $\dot{f}(\mathbf{r},u)$ is independent of the electric field in the medium. For example if the decay function $f(\mathbf{r},u)$ is just a step function (from zero) at $u = t-t' = 0$ then equations (19) and (20) reproduce an instantaneous correlation between the electric and displacement fields. In the next section we will discuss the implications of these physical constraints upon dielectric permittivity behavior.

In summary, a universal electric displacement field or '**D**-field' can be defined which is inclusive of all charges. This generalized **D**-field is necessary for use in dielectric composites. Provided there are no unaccounted for sources, sinks, or charge accumulations, then this field is divergenceless and continuous. A normalized displacement decay current/function can be introduced to statistically relate the present observed displacement field to the past electric field history.

DIELECTRIC PERMITTIVITY

Dielectric permittivity is defined by a tensor relationship between the electric and displacement field vectors in the frequency domain which may be expressed as

$$\mathbf{D}(\mathbf{r},f) = \epsilon(\mathbf{r},f) \cdot \mathbf{E}(\mathbf{r},f) \tag{21}$$

where $\mathbf{E}(\mathbf{r},f)$ and $\mathbf{D}(\mathbf{r},f)$ are respectively the electric and universal displacement fields at a location $\mathbf{r}$ and frequency f. Alternately this relationship can be expressed using spatial index notation

$$\mathrm{D}_i(r,f) = \epsilon_{ij}(\mathbf{r},f)\ \mathbf{E}_j(\mathbf{r},f)\ . \tag{22}$$

The dielectric permittivity $\epsilon_{ij}(r,f)$ is a tensor of rank two.

The dielectric permittivity is defined in the frequency regime because, strictly speaking, there are no known substances

(except vacuum) which exhibit complete instantaneous displacement response to the application of an electric field. The implication of this fact is that there are no completely static time domain constants between electric and displacement fields. Rather there exists a time correlation behavior between the present displacement field and the past applied electric fields. This is expressed by equations (19) and (20) and the normalized displacement decay current $\dot{f}(\mathbf{r},t-t')$ or through its accumulation as the normalized displacement decay function $f(\mathbf{r},t-t')$. The frequency domain transformation of the convolution relation expressed by equations (19) and (20) produces the needed relation connecting the frequency domain permittivity $\epsilon(\mathbf{r},f)$ with the normalized displacement decay current $\dot{f}(\mathbf{r},t-t')$ of the time domain as

$$\epsilon(\mathbf{r},f) = \int_0^\infty \dot{f}(\mathbf{r},u)\ e^{+j2\pi fu}\ du \tag{23}$$

where the time variable is $u = t-t'$. Because the decay current and decay function are real valued, the frequency domain permittivity $\epsilon(\mathbf{r},f)$ is necessarily complex valued unless the decay function is instantaneous. Traditionally, in the field of dielectrics, this is expressed as

$$\epsilon(\mathbf{r},f) = \epsilon'(\mathbf{r},f) + j\ \epsilon''(\mathbf{r},f) \tag{24}$$

where $\epsilon'(\mathbf{r},f)$ and $\epsilon''(\mathbf{r},f)$ are respectively the real and imaginary parts of the dielectric permittivity. The imaginary part of permittivity is known as the "lossy" part as it is proportional to the energy lost during a cycle of a time harmonic field.

When a medium is isotropic, both the displacement and electric field vectors are colinear and the permittivity can be regarded as the scalar ratio $\epsilon(\mathbf{r},f)$ of the fields $D(\mathbf{r},f)/E(\mathbf{r},f)$. In situations where the medium is anisotropic or birefringent, then the fields are reoriented locally until they lie along the direction of a principal axis of the local permittivity tensor. In this alignment the fields are colinear and their ratio can be measured as before as the corresponding element in the diagonalized permittivity tensor. In the general anisotropic situation where the principal directions may not experimentally be possible to determine, measurements have to be made of all spatial aspects pertaining to both field vectors. For isotropic dielectrics where the permittivities are scalar and spatially uniform, the ratio of the imaginary to real part of the permittivity is called the loss tangent or dissipation factor [24, 25]. This is conventionally written as

$$\epsilon''(\mathbf{r},f)/\epsilon'(\mathbf{r},f) \equiv \tan\ \delta(\mathbf{r},f)\ . \tag{25}$$

In this report we will examine the case of random (Monte Carlo) composite media simulations with a high degree of macroscopic isotropy. Expansion to anisotropic cases is anticipated in future work.

As an illustrative case, when the decay function is isotropic and spatially uniform with an exponential decay $f(t) = \epsilon_A (1 - e^{-t/T})$ (reflecting a viscous type relaxation), it transforms into the familiar simple form of Debye relaxation

$$\epsilon(f) = \int_0^\infty (\epsilon_A/T)(e^{-t/T})(e^{+j2\pi fu})\, du \tag{26}$$

$$\epsilon(f) = \epsilon_A/(1 - j2\pi fT) \tag{27}$$

where ϵ_A is the transition permittivity (a constant), and T is the viscous relaxation time. Note that $\epsilon(f)$ here is a complex numbered quantity expressible as the real and imaginary pair

$$\epsilon'(f) = \epsilon_A/[1 + (2\pi fT)^2] \tag{28}$$

and

$$\epsilon''(f) = \epsilon_A\, 2\pi fT/[1 + (2\pi fT)^2] \;. \tag{29}$$

An interesting feature of the Debye relaxation permittivity is that the complex plane plot of $\epsilon'(f)$ and $\epsilon''(f)$ reveals a semicircle as frequency is varied. This type of plot is often referred to as the Cole-Cole plot and is useful as a way of identifying the relaxation as well as fingerprinting permittivity characteristics. Elimination of the explicit frequency variable on the right hand sides of the above equation pair does indeed verify a semicircle of radius $\epsilon_A/2$ whose imaginary part reaches its maximum value when the frequency is at $1/2\pi T$.

CONDUCTIVITY

A universal displacement conductivity $\boldsymbol{\sigma}(\mathbf{r},f)$ is defined by a tensor relationship between the electric field and universal displacement current density vectors in the frequency domain as

$$\mathbf{J}_D(\mathbf{r},f) = \boldsymbol{\sigma}(\mathbf{r},f)\cdot\mathbf{E}(\mathbf{r},f) \;. \tag{30}$$

Combining equations (18), (21), and (30) yields

$$\boldsymbol{\sigma}(\mathbf{r},f)/j2\pi f = \boldsymbol{\epsilon}(\mathbf{r},f) \tag{31}$$

as the connection of the universal displacement conductivity and the universal permittivity subject to the constraint that $f \neq 0$. The convolution for universal displacement conductivity becomes

$$\boldsymbol{\sigma}(\mathbf{r},f) = j2\pi f \int_0^\infty \mathbf{f}(\mathbf{r},u)\; e^{+j2\pi fu}\, du \;. \tag{32}$$

The universal displacement conductivity as a complex number can be expressed in terms of its real and imaginary parts as

$$\boldsymbol{\sigma}(\mathbf{r},f) = \boldsymbol{\sigma}'(\mathbf{r},f) - j\, \boldsymbol{\sigma}''(\mathbf{r},f) \tag{33}$$

where $\boldsymbol{\sigma}'(\mathbf{r},f)$ and $\boldsymbol{\sigma}''(\mathbf{r},f)$ are respectively the real and imaginary parts. The direct current (DC) conductivity is the limiting case

of the real part of the conductivity when the frequency tends toward zero. However the projected value of DC conductivity becomes less distinct when nonstatic measurements are made at higher and higher frequencies especially over a limited bandwidth.

Exploiting the identity posed by equation (31) yields

$$\boldsymbol{\sigma}'(\mathbf{r},f) = 2\pi f \; \boldsymbol{\epsilon}''(\mathbf{r},f) \tag{34}$$

and

$$\boldsymbol{\sigma}''(\mathbf{r},f) = 2\pi f \; \boldsymbol{\epsilon}'(\mathbf{r},f) \; . \tag{35}$$

When a medium is isotropic, the displacement current density $\mathbf{J}_D(\mathbf{r},f)$ and electric field $\mathbf{E}(\mathbf{r},f)$ vectors are colinear. In this instance, the universal displacement conductivity can be regarded as the scalar ratio $\sigma(\mathbf{r},f)$ of vectors given by $\mathbf{J}_D(\mathbf{r},f)/\mathbf{E}(\mathbf{r},f)$. If the medium is anisotropic or birefringent, then one may reorient alongside a principal axis of the local conductivity tensor to determine the tensor components. The loss tangent defined in equation (25) becomes

$$\tan \delta(\mathbf{r},f) = \sigma'(\mathbf{r},f)/\sigma''(\mathbf{r},f) \; . \tag{36}$$

in terms of the real and complex conductivity parts.

CAUSALITY

A physical response is said to be causal if it occurs at or following an excitation. In classical systems most responses cannot anticipate future excitations and hence their connecting functions are null. A partial relaxation of this behavior may occur in quantum mechanical systems wherein the connecting function becomes a probability. Quantum mechanical causality or other non-local overlapping shall not be dealt with in this report. Both frequency domain universal dielectric permittivity $\boldsymbol{\epsilon}(\mathbf{r},f)$ and conductivity $\boldsymbol{\sigma}(\mathbf{r},f)$ arise from a causal function. This function is the time domain normalized displacement decay current $\dot{\boldsymbol{f}}(\mathbf{r},u)$ or alternately, its accumulation, the normalized displacement decay function $\boldsymbol{f}(\mathbf{r},u)$. The time variable u is the difference between present and past times u = t-t'. The decay function is a time domain function and it must be zero when u is negative (a non-zero value would indicate future excitations that cannot have happened yet). The decay function relates real-valued physical vector quantities over past history as specified in the convolution equations (19) and (20). The consequence is that the decay function and permittivity are related by transformation equation (23). The inverse relation that obtains the decay function from the permittivity spectrum exists and is required to possess the aforementioned physical constraints. The inversion procedure is accomplished by means of complex Laplace transforms or equivalently unilateral/one-sided Fourier transforms. This procedure requires that the frequency be

treated as a complex number which can be viewed as lying in the complex frequency ('s') plane. The complex frequency is defined as

$$s \equiv -j2\pi f \ . \tag{37}$$

The substitution of the variable s into equation (23) leads to the alternate expression for the permittivity

$$\epsilon(\mathbf{r},s) = \int_0^\infty \boldsymbol{f}(\mathbf{r},u)\ e^{-su}\ du \tag{38}$$

where the other variables are the same as before. In Laplace transform operator notation equation (32) can be succinctly written as

$$\epsilon(\mathbf{r},s) = \mathcal{L}(\boldsymbol{f}(\mathbf{r},u)) \tag{39}$$

where $\mathcal{L}$ denotes the Laplace integral transform operation $\int_0^\infty [\ldots]\ e^{-su}\ du$ acting upon $\boldsymbol{f}(\mathbf{r},u)$. The inverse operation can be obtained through residue theory and is

$$\boldsymbol{f}(\mathbf{r},u) = \frac{1}{2\pi j} \int_{+c-j\infty}^{+c+j\infty} e^{+su}\ \epsilon(\mathbf{r},s)\ du \tag{40}$$

where the contour c is chosen such that all the singular points of $\epsilon(\mathbf{r},s)$ lie to the left of the contour on the s-plane. In Laplace operator notation one can simply write equation (40) as

$$\boldsymbol{f}(\mathbf{r},u) = \mathcal{L}^{-1}(\epsilon(\mathbf{r},s)) \tag{41}$$

where $\mathcal{L}^{-1}$ denotes the inverse Laplace transform as given in the expression (39).

The significance of these transformation expressions between the permittivity and decay function is that of two important properties: 1) complex conjugation, and 2) analyticity and interdependence between real and imaginary parts. These two properties are of consequence to dielectric observation.

1) The property of complex conjugation requires that the permittivity (as well as conductivity) become complex conjugate whenever the s-frequency becomes conjugate, i.e.

$$\epsilon^*(\mathbf{r},s) = \epsilon(\mathbf{r},s^*) \ , \tag{42}$$

where the asterisk superscript denotes conjugation of the preceding variable. The relationship expressed by equation (42) can be inferred by complex conjugation of the s-frequency in the transformation relations, (38) through (41). Graphically, on the s-plane the real parts of the permittivity and conductivity functions are mirror symmetric about the real s-axis, while the imaginary parts are antisymmetric about the real s-axis. This property not only allows complex conjugation to occur between

physicists and electrical engineers, i.e., $j = -i$, but also applies whenever a dielectric permittivity may be made up of functions which are functions of complex frequency. This occurs, for example, in the case of composite mixtures. Therefore the action of applying composite mixture relations to well-behaved constituents cannot introduce any violations of the conjugation property in the resultant dielectric response.

2) The second property of analyticity and real/imaginary parts interdependence is commonly referred to as the Kramers-Kronig relationship [23, 26-28]. This relationship can be stated in different forms such as a pair of complementary Hilbert transforms or as a pair of one-sided integrals between the real and imaginary parts of permittivity or conductivity. The Kramers-Kronig relationship is manifested in the analyticity (i.e., Cauchy-Riemann conditions) and lack of singularities of the permittivity function on the right side (positive s-values, real part) of the s-plane. If a singularity occurs on this portion of the s-plane, then an unstable or undefined dielectric system response would occur at some range of physically realizable excitation frequencies. In order that the decay function $\boldsymbol{f}(\mathbf{r},u)$ consistently remains causal and real valued, then the singularities can only exist on the left side of the s-plane either on the negative real s-axis or as complex conjugate pairs on the left side (negative s-values, real part) of the s-plane. The Kramers-Kronig relation for permittivity may be written

$$\epsilon(\mathbf{r},s) = \frac{-j}{\pi} ⨍_c \frac{\epsilon(\mathbf{r},z)}{z-s}\, dz \tag{43}$$

where the integral is principal valued with the contour c running first along the imaginary s-axis from negative to positive then clockwise in a large semicircle about the right half s-plane. The variable z is a variable of integration. Using the complex conjugation property of equation (42) and reverting back to the ordinary frequency notation f, the Kramers-Kronig relation can be written as the integral pair

$$\epsilon'(\mathbf{r},f) = \frac{2}{\pi} \int_0^\infty \frac{f'\ \epsilon''(\mathbf{r},f')}{(f'^2-f^2)}\, df' \tag{44}$$

and

$$\epsilon''(\mathbf{r},f) = \frac{2f}{\pi} \int_0^\infty \frac{\epsilon'(\mathbf{r},f')}{(f'^2-f^2)}\, df' \ . \tag{45}$$

Kramers-Kronig requirements also apply to composite mixture permittivities in that the mixing relations cannot introduce contradictions to well-behaved constituents.

MACROSCOPIC DIELECTRIC QUANTITIES

Both composite dielectric analysis and dielectric data acquisition are carried out over regions of finite spatial extent. Dielectric data acquisition can be a formidable task because of a limited ability to resolve the electric and displacement field vector components, as well as other considerations such as extraneous polarizations and stray fields.

Macroscopic quantities must be introduced such that the overall displacement flux remains continuous. The conversion can be accomplished in going from the differential form equation (10) (Gauss' law) into an integral form via the divergence theorem of mathematics as

$$\int_v \boldsymbol{\nabla}\cdot\mathbf{D}(\mathbf{r},t)\ dr^3 = \int_v 0\ dr^3 \tag{46}$$

and

$$\oint_a \mathbf{D}(\mathbf{r},t)\cdot\hat{\mathbf{n}}\ dr^2 = 0\ . \tag{47}$$

In equation (46), the integration is performed over the volume v enclosed by a closed surface. In equation (47), the integration is taken over the surface enclosing volume v, and $\hat{\mathbf{n}}$ is the outward drawn surface normal unit vector. We can conveniently restate equation (47) in terms of a universal displacement flux Φ corresponding to the integrated displacement field passing through a given surface. This may be written in the time domain as

$$\Phi_n(t) \equiv \int \mathbf{D}(\mathbf{r},t)\cdot\hat{\mathbf{n}}\ dr^2 \tag{48}$$

or in the frequency domain as

$$\Phi_n(f) \equiv \int \mathbf{D}(\mathbf{r},f)\cdot\hat{\mathbf{n}}\ dr^2 \tag{49}$$

with the integral taken over the area of the surface in question. In terms of displacement flux Φ, the integral form of equation (47) becomes

$$\oint_a d\Phi(\mathbf{r},t) = 0 \tag{50}$$

in the time domain or

$$\oint_a d\Phi(\mathbf{r},f) = 0 \tag{51}$$

in the frequency domain. In equations (50) and (51), $d\Phi(\mathbf{r},t)$ and $d\Phi(\mathbf{r},f)$ denote the incremental displacement flux as the integration proceeds around the closed surface. The right hand sides of equations (50) and (51) are zero for this universal form of electric displacement because we have chosen the displacement to be inclusive of all charges and an overall charge neutrality exists within the enclosure. The units of displacement flux Φ are that of charge. The time derivative of displacement flux

possesses units of charge current. Equations (50) and (51) are thus simply a statement of charge/displacement field continuity and conservation. The interpretation of an electric displacement field flux is that a flux originates or terminates on a charge of that value.

The conversion of equations (10) and (16) into the integral equations (50) and (51) requires the choice of some enclosing 'Gaussian' surface. Generally this choice is that of convenience over which the macroscopic permittivity $\boldsymbol{\epsilon}(f)$ is defined. The macroscopic permittivity is defined in the same sense as the microscopic permittivity of equations (21) and (22) with the distinction that the field quantities are mean valued over the volume of the enclosure. The defining relation for the macroscopic permittivity tensor $\boldsymbol{\epsilon}(f)$ becomes

$$\mathbf{D}(f) = \boldsymbol{\epsilon}(f)\cdot\mathbf{E}(f) \tag{52}$$

where $\mathbf{D}(f)$ and $\mathbf{E}(f)$ are respectively the mean valued displacement and electric fields over the selected 'Gaussian' enclosure. Since the permittivity or conductivity is always defined in the context of the frequency domain, for the remainder of this report we will drop the explicit frequency dependence notation with the understanding that the frequency dependence does remain. Thus in this reduced shorthand notation the permittivity $\boldsymbol{\epsilon}$ is

$$\mathbf{D} = \boldsymbol{\epsilon}\cdot\mathbf{E} \tag{53}$$

where again $\mathbf{D}$ and $\mathbf{E}$ are respectively the displacement and electric fields understood to be macroscopic and of the frequency domain. The macroscopic versions of other relevant dielectric relations are as follows:

i) The evolution from a decay function corresponding to equation (23) is

$$\boldsymbol{\epsilon} = \int_0^\infty \mathbf{f}(u)\ e^{+j2\pi fu}\ du \tag{54}$$

where $\boldsymbol{\epsilon}$ is the macroscopic permittivity (frequency dependent) and $\mathbf{f}(u)$ is the normalized macroscopic displacement decay current. The integration is performed over all past times u at fixed frequency f.

ii) The macroscopic permittivity is expressible in terms of its real and imaginary parts as in equation (24) as

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}' + j\ \boldsymbol{\epsilon}'' . \tag{55}$$

iii) The loss tangent for the macroscopic and isotropic dielectric specimen is the ratio of the imaginary to real permittivity portions

$$\tan\delta \equiv \epsilon''/\epsilon' . \tag{56}$$

iv) The displacement current conductivity in the macroscopic case is found from equation (30) to be

$$\mathbf{J}_D = \boldsymbol{\sigma} \cdot \mathbf{E} \tag{57}$$

where $\mathbf{E}$ is the electric field and $\mathbf{J}_D = \mathbf{D}/ t$ is the universal displacement current density at a given frequency.

v) The relation between the macroscopic permittivity and conductivity is found from equation (31) to be

$$\boldsymbol{\sigma}/j2\pi f = \boldsymbol{\epsilon} \ . \tag{58}$$

vi) The macroscopic conductivity can be separated into its real and imaginary parts as

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}' - j\,\boldsymbol{\sigma}'' \ . \tag{59}$$

vii) When the medium is isotropic with respect to the macroscopic permittivity, it also must be isotropic in the displacement conductivity and thus the loss tangent given by

$$\tan\,\delta = \sigma'/\sigma'' \tag{60}$$

can be found in similar fashion to that of equation (36).

viii) A macroscopic medium obeys causality and its behavior is derivable as an analytic function through the use of Laplace transform techniques upon the normalized displacement decay current. In Laplace operator notation we can write

$$\boldsymbol{\epsilon}(s) = \mathcal{L}(\boldsymbol{\mathfrak{f}}(u)) \tag{61}$$

where $\mathcal{L}$ denotes the Laplace integral transform operation $\int_0^\infty [\ldots]\, e^{-su}\, du$ acting upon $\boldsymbol{\mathfrak{f}}(u)$.

ix) The complex conjugation property demands that a causal analytic function which stems from a real valued time function obey

$$\boldsymbol{\epsilon}^*(s) = \boldsymbol{\epsilon}(s^*) \ . \tag{62}$$

The complex conjugation property imposes a restriction on the choice of composite mixture formulae. Application of this property to the k^{th} constituent species and considering that the composite response also must be causal results in

$$\boldsymbol{\epsilon}^* = \mathfrak{f}\{\boldsymbol{\epsilon}_1{}^*, \boldsymbol{\epsilon}_2{}^*, \boldsymbol{\epsilon}_3{}^*, \ldots \boldsymbol{\epsilon}_k{}^*, \ldots\} \tag{63}$$

where $\boldsymbol{\epsilon}^*$ is the conjugate permittivity of the composite, $\boldsymbol{\epsilon}_k{}^*$ is the conjugate permittivity of the k^{th} constituent species, and $\mathfrak{f}$ denotes some functional.

x) The Kramers-Kronig relation expresses the interdependence

between the real and imaginary parts as

$$\epsilon(s) = \frac{-j}{\pi} \int_c \frac{\epsilon(z)}{z-s} dz \qquad (64)$$

where the integral is principal valued with the contour c running first along the imaginary s-axis from negative to positive then clockwise in a large semicircle about the right half s-plane. The variable z is a variable of integration. Such a relationship is of use in determining valid composite mixture formulae and for checking authenticity or filling in data gaps.

The interrelations expressed or implied in i) through x) for a macroscopic dielectric specimen are valid providing the macroscopic 'Gaussian' enclosure boundaries do not change.

DIELECTRIC COMPOSITE MIXTURE FORMULAE

Knowledge of the dielectric characteristics such as permittivity for a particular macroscopic configuration does not imply full knowledge of the subassembly of possible microscopic configurations. This degeneracy exists whenever a dielectric medium is nonuniform such as in the case of composites. This degeneracy even already exists for a simple composite constructed of stratified flat layers with fixed constituent volume fractions. In this special case, the solution can be worked out with treatment as a collection of capacitors aligned either in series or parallel [4, 29].

As a consequence, there can exist a set of composite mixture formulae which will satisfy or nearly satisfy a given particular macroscopic dielectric observation. Evidence of such degeneracy of formulae can be found in some of reviews on composite mixtures [25, 30-34]. Moreover, the wavelengths of the probing fields typically applied in dielectric measurements generally are not capable of resolving microscopic features and therefore the measured macroscopic response can only reflect collective behaviors.

One criterion for the applicability of a particular composite mixture formula is that the dielectric response predicted for a composite cannot introduce any new information, particularly in regard to microscopic configurations. In the case of random composites there is by definition a lack of specific knowledge of microscopic configurations and in order for a mixture formula to be of relevance it must contain a minimum of parameters pertaining to the microscopic configuration. The composite dielectric response predicted by a mixture formula must also reflect the same symmetry properties that are posed by the physical input situation.

Two mixture formulae can be shown to be particularly applicable to the case of random dielectric composites, namely

Lichtenecker's formula as given in equation (1) and the EMT-CPA formula as given in equation (2). Each of these two composite mixture formulae can be explained with reference to their constraints and applicability, as follows:

I) The Lichtenecker's dielectric mixture formula of equation (1) is a self-similarity formula derivable by a process of constraint elimination starting with a generalized expression [5, 6]. The several constraining arguments are:

1) Proportionality and role symmetry must be maintained between the resultant outcome and that obtained when all the constituents are changed by the same common factor. Mathematically this may be stated as

$$m\epsilon = g[m\epsilon_1, m\epsilon_2, m\epsilon_3, \ldots] \tag{65}$$

where m is a real multiplicative factor, g means a generalized functional, ϵ is the composite permittivity, ϵ_1 is the permittivity of constituent species 1, etc.. The role symmetry argument implies that no one constituent be different than any other as far as its contribution to the overall composite dielectric behavior.

2) Mixture responses must be invariant with further random mixing of the type that was employed in order to attain its present state. In other words, if a truly random state has been reached then further 'stirring' does not affect the response of the composite. This property can also be stated as additive functional invariance to mixing by successive stages of mixtures of mixtures. This relation would be

$$f(\epsilon) = \sum_k v_k \, f(\epsilon_k) \tag{66}$$

where f is a common functional, ϵ is the composite permittivity, ϵ_k is the permittivity of the k^{th} constituent species occupying a volume fraction v_k, and the summation $\sum_k$ is over the k species. This property assumes that constituent volume densities are not affected by successive mixing stages.

3) The macroscopic dielectric response must be independent of the sample size considered. This results in the mixture response being dependent on relative volume fractions and not overall macroscopic size.

4) The constituent volume densities must remain constant during the mixing process. This requirement results in the additive relation

$$v_k = \sum_j V_j \, v_{j,k} \tag{67}$$

where v_k is the fractional volume of the k^{th} constituent species in the present stage of macroscopic volume consideration, V_j is the volume fraction of a submacroscopic volume that goes into the

present macroscopic volume, and $v_{j,k}$ is the subvolume fraction of the k^{th} species within the j^{th} submacroscopic volume. The summation is exhaustive of all possible j^{th} submacroscopic volume fractions. Presumably, the constituents cannot interact with each other so as to affect the volume density of the other.

The Lichtenecker mixing formula is a self-similarity formula of the type found in references [35-38], meaning that regardless of the macroscopic size scale chosen for observation, one can expect the same result over and over again. In actuality, natural materials may only partially fulfill these requirements, especially if the random mixing occurs only at a particular size scale. Such a violation is evident when considering molecularly interdispersed mixtures, even when otherwise perfectly randomly distributed with no overall sequential ordering.

The exponential averaging factor α of the Lichtenecker mixing formula must be real valued if the mixture is to be causal and obey the complex conjugation property as given by equations (62) and (63). The exact value of the exponential averaging factor α has not been entirely ascertained for random mixtures. The value of the exponential averaging factor must lie within a set of physical bounds called the Wiener bounds. These bounds require that the exponential averaging factor must be real valued at or between minus/plus unity. The value of the exponential averaging factor that is found in a given random composite configuration will depend on spatial degrees of freedom and percolation path(s) available to the displacement fluxes or currents as they traverse the assortment of domain regions. If the composite is built up from submixtures all of the same exponential averaging factor then the composite has the same exponential averaging factor. This then implies the exponential averaging factor has a constant value for a particular mixture type. When the permittivities of the constituents are close to each other then the exponential averaging factor α takes on limiting values which tend to some of the depolarization values which are discussed next.

II) The effective-medium-theory coherent-potential-approximation (EMT-CPA) has been derived and rederived many times [13-15, 39-42]. This approximation is one of the simpler of effective medium theory. Fundamentally, its argument is that a domain region containing one of the constituent permittivities is treated as being surrounded by a medium whose effective permittivity is to be determined. All other domains are treated accordingly with the same approximation of environment. As a further approximation a certain domain geometry is presumed, so as to lend to analytic solution of the electric and displacement fields. The geometry selected is that of ellipsoids, as both Maxwell's electromagnetic equation set can be solved using a conformal coordinate system [16, 17], and because this geometry involves a minimum of structural detail. This treatment corresponds to the lowest order scattering of the solution of Maxwell's electromagnetic equations in an inhomogeneous media

[15, 43]. This treatment is called the coherent potential approximation. A number of other approximations and refinements can be made using the effective medium technique [32, 44, 45]. However, the approach appropriate to random composites must introduce a minimum of microscopic detail and it is felt the coherent potential approximation does offer such. Further subtreatments do exist in so far as different types of randomness can exist in composites.

An important factor which arises from the effective-medium-theory coherent-potential-approximation (EMT-CPA) approach is the depolarization factor 'A'. The depolarization factor 'A' can be calculated using an integral formula which depends on the ellipsoid shape and is discussed in greater detail in Appendix A [16-18, 32]. In the case of a spheroid situated in N spatial dimensions, the depolarization is simply the inverse of the number of spatial dimensions (degrees of freedom) as

$$A = 1/N \ . \tag{68}$$

When constituent permittivities are close valued with respect to each other then the exponential averaging factor can be related to the depolarization as

$$\alpha \rightarrow 1 - 2A \ . \tag{69}$$

Combining the limit in equation (69) with the identity of equation (68) yields the limiting parameter value

$$\alpha >\rightarrow 1-2/N \tag{70}$$

when the constituent permittivity values are close valued with respect to each other.

In this report we treat both the exponential averaging factor and depolarization as statistical parametric values. Our results are presented on a parametric basis both in terms of the exponential averaging factor 'α' and the depolarization factor 'A' for both clarity of permittivities on relative scales, and presentation of complex numbered permittivities in terms of real valued parameters.

PERCOLATION

The term percolation refers to whether or not there exists a continuous connected path through a constituent species. In the context of the dielectric problem, percolation is whether or not displacement flux/current has at least one path through connected domains of a single constituent species. A permittivity/conductance of zero blocks electric displacement and an infinite permittivity/conductance allows displacement flux/current to pass freely along a path. The situation where the displacement flux/current is in a single constituent but not the other occurs as the limiting case where the permittivities

are of infinite ratio to one another and there exists a percolation path amongst the larger permittivity domains.

The Weiner bounds (i.e., the series and parallel stratification limits for dielectric composites) can be used to represent the two extremes of percolation behavior. If the constituents lie in flat layers with surface normals oriented parallel to the applied electric field then the exponential averaging factor as defined by equation (1) takes on the value $\alpha = -1$. If the species lie in flat layers with surface normals oriented perpendicular to the electric field then the exponential averaging parameter is $\alpha = +1$. For the depolarization as defined by equation (2), direct substitution of the parameter values A=1 and A=0 give respectively the series and parallel stratification limits. Thus the depolarization is also a measure of percolation with respect to the Wiener bounds.

Most often percolation is discussed in the context of binary mixtures in which one constituent is nonconducting and the other is fully conducting. It has been found that as the volume fraction of the conducting constituent increases, a transition occurs once a conducting path has been established throughout the composite mixture. This transition is called the percolation threshold. At the percolation threshold, network simulations with iso-sized shuffling elements become self-similar or scaleless [38]. At other constituent mixing ratios this self-similarity may not hold. Lichtenecker's mixture formula stipulates that a self-similarity pattern must always be maintained in a mixture type regardless of constituent mixing ratios. The network simulations used in this report principally employ iso-sized shuffling elements but do not exhibit self-similarity except at percolation threshold.

Percolation mixtures have been extensively studied [38, 46, 47]. It has been found that two-dimensional mixtures with a random sputtering of iso-sized grains have critical percolation transitions at .50 for bond cubic lattices and .59 for site cubic lattices. Our studies of the exponential averaging factor tend to confirm the percolation threshold behavior for conducting/nonconducting binaries. At the percolation threshold the Lichtenecker factor achieves its best self-similarity values.

THE MACROSCOPIC SELECTION

The 'Gaussian' enclosure used in this report is a rectangular box. This choice lends itself to a simple formulation for the macroscopic quantities as well as being easy to envision as a probe parallel-plate capacitor with no fringing fields. The electric field is applied between two opposite faces of the box contacting normally to these surface boundaries. In essence these surfaces form the plates of the probe capacitor. The mean value of the quasistatic electric field is normal to the plate surfaces and has a magnitude of the potential difference divided by the gap distance separating the plates. In numerical

terms this is

$$E = -V/d \tag{71}$$

where E is the mean electric field magnitude, V is the potential difference, and d is the gap distance separating the plates. The mean value of the displacement field is determined by the displacement flux density averages passing through each of the three opposite face pairs. Thus for each face the mean displacement field can be written as

$$D_n = \Phi/a \tag{72}$$

where D_n is the magnitude of the mean displacement field component normal to the face, Φ is the displacement flux passing through a face, and 'a' is the surface area of the face. One of the opposite face pairs are the probe capacitor plates. The other two opposite face pairs are termed as lateral faces. The permittivity tensor can be determined using equation (52). The probe capacitor may be oriented along any direction, but for convenience it may be oriented along a principal axis whereupon the electric and displacement fields are colinear. In the isotropic case the displacement and electric field are always colinear which implies that regardless of the orientation there is no lateral displacement flux. We used this fact as a test of isotropy.

The experimentalist may also choose to use a probe capacitance in the comparative sense in isotropic cases. That is

$$\epsilon = \epsilon_o \ (C/C_o) \tag{73}$$

where ϵ is the unknown permittivity, ϵ_o is the reference permittivity, C is the measured capacitance with dielectric, and C_o is the reference capacitance.

COMPOSITE DIELECTRIC STRUCTURE

A dielectric composite is a spatial aggregation of interspersed and interconnected permittivity domains. In this report, we seek to calculate the resultant macroscopic permittivity of a composite material for the case of composites which have insignificant quantum overlap between domains. The resultant permittivity can be analyzed in relation to that predicted by the mixture formulae of equations (1) and (2); the results of the numerical simulations will be displayed in terms of the exponential averaging factor 'α' and depolarization factor 'A'.

The dielectric composite is solved in terms of a pixel-like rectangular grid as shown in Figure 1. A domain is made up of one or more pixels. Each pixel region is then assigned the permittivity of the domain it represents. Each pixel experiences a local electric field and in turn responds with a displacement

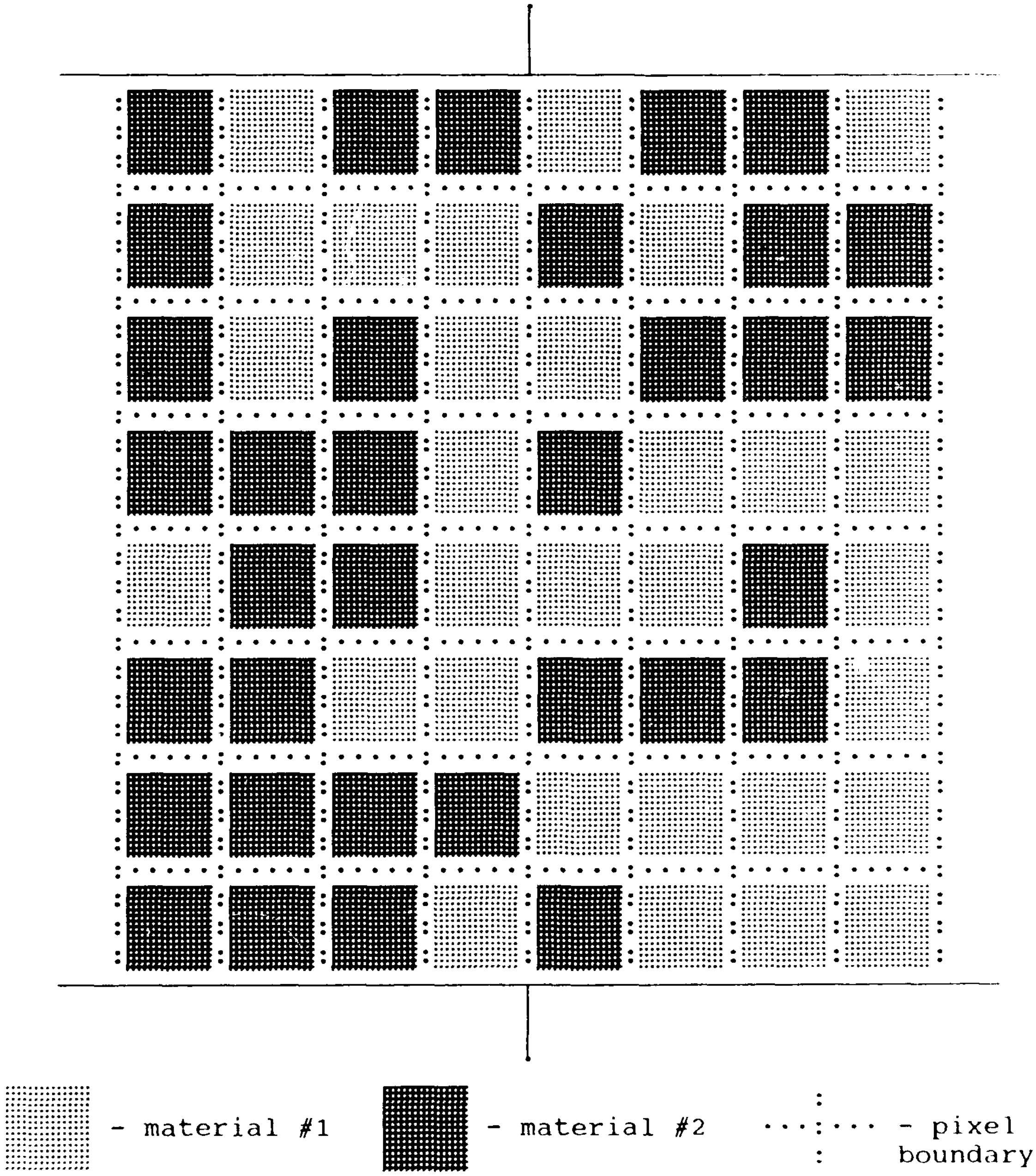


Figure 1. Cross-sectional view of the probe parallel-plate capacitor 'Gaussian' enclosure used in this report. Within the sample region, the composite dielectric is represented by a rectangular pixel arrangement.

field as determined by its assigned permittivity. Displacement field continuity is preserved in passage amongst the pixels. In the general case of a random arrangement of domains there can be strong effects due to local fields when the permittivity

differences are large between the constituents. Even though the macroscopic response tends to wash out local or microscopic details, these small details seem to matter when the mixture is near percolation and the constituent permittivity differences are large. Because of this effect, the numerical analysis requires a fairly high degree of detail or one must somehow cover the crucial details. The approximation employed in this report is a method which is sensitive to at least some degree to local effects.

The pixels are assigned the permittivity value of one of the constituent materials in order not to lend a composite mixing bias to the overall solution, since to assign a pixel a permittivity other than one of the constituent permittivities would require some sort of assumption about possible composite mixing within the pixel region. Such an assumption can only be made once some composite mixture relationship is established.

FINITE DIFFERENCE OR NETWORK ANALYSIS

Equations (10) and (21) may be solved simultaneously and macroscopically to any accuracy using network analysis or equivalently finite difference approximation. The procedure involves the division of the macroscopic sample into a mesh or node set of macroscopic subregions. By introducing a regular rectangular mesh of points $\{\mathbf{r_i}\}$ with spacings δr_{ij} to contacting neighbors one obtains a system of linear equations at each i^{th} mesh point with contacting j^{th} neighbors

$$\sum_j \Phi_{ij} = 0 \tag{74}$$

or equivalently

$$\sum_j I_{Dij} = 0 \tag{75}$$

where Φ is the universal displacement flux and $I_D = d\Phi/dt$ is the universal displacement current. The summation is over the j^{th} contacting neighbors of the i^{th} point, and mesh points i and j contain the entire address information needed to specify each mesh point. The displacement flux and current are each determined by the displacement field or displacement current density normal to the interface surface areas surrounding each mesh point. The determination of these quantities is performed as described previously in the discussion of macroscopic dielectric quantities. Expanding equations (74) and (75) one obtains

$$\sum_j D_{ij}\, \delta a_{ij} = 0 \tag{76}$$

and

$$\sum_j I_{Dij}\, \delta a_{ij} = 0 \tag{77}$$

where D_{ij} or its time derivative $dD_{Dij}/dt = I_{Dij}$ are respectively

the mean valued displacement field or current normal to surface δa_{ij}, and δa_{ij} is the surface area common to the Gaussian enclosures between the i^{th} and j^{th} nodes. The local displacement field may be related to the local electric field as

$$\mathbf{D}_{ij} = \boldsymbol{\epsilon}_{ij}\,\mathbf{E}_{ij} \tag{78}$$

where $\mathbf{D}_{ij}$ and $\mathbf{E}_{ij}$ are the mean valued displacement and electric fields on the interface between the i^{th} and j^{th} nodes, and $\boldsymbol{\epsilon}_{ij}$ is the permittivity characteristic between the i^{th} and j^{th} nodes. The mean electric field can be expressed quasistatically as a potential difference

$$E_{ij} = \delta V_{ij}/\delta r_{ij} \tag{79}$$

where E_{ij} is the mean electric field, $\delta V_{ij} = (V_j - V_i)$ is the potential difference between nodes, and $\delta r_{ij} = |\mathbf{r}_j - \mathbf{r}_i|$ is the internodal separation. The potential or voltage at the i^{th} mesh node point is denoted as V_i. The electric field E_{ij} is the component in the direction perpendicular to the interface shared commonly between the i^{th} and j^{th} nodes. Since rectangular boxes are selected in this report as the type of macroscopic enclosure, the surface normal direction lies along the same direction as the internode gaps $\delta\mathbf{r}_{ij}$ when the nodes are placed at the box centers. Substituting the equations (78) and (79) into equation (76) produces at each i^{th} node the sum over the j^{th} neighbors

$$\sum_j (\epsilon_{ij}\,\delta V_{ij}\,\delta a_{ij})/\delta r_{ij} = 0 \tag{80}$$

where the notations are as before. The potential drops, $\delta V_{ij} = (V_j - V_i)$, are the unknowns in a system of simultaneous equations formed when all nodes are taken together with an exciting electric field or potential applied across the probe. The factors other than the potential drops in equation (80) may be combined into a set of internodal admittances, yielding

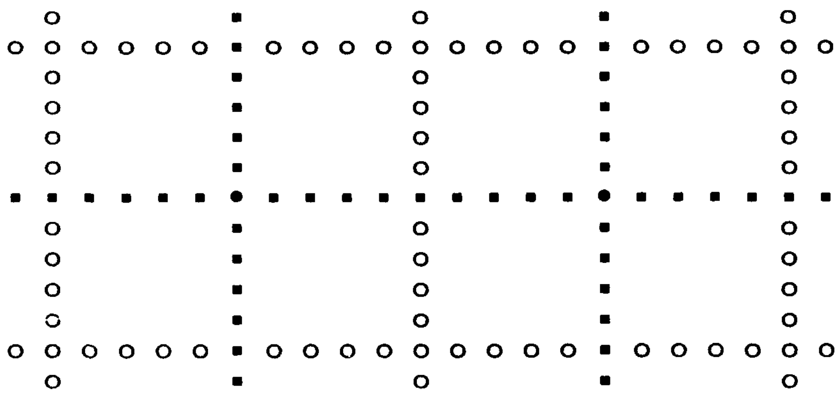
$$\sum_j g_{ij}\,\delta V_{ij} = 0 \tag{81}$$

with the internodal admittances just $g_{ij} = (\epsilon_{ij}\,\delta a_{ij}/\delta r_{ij})$.

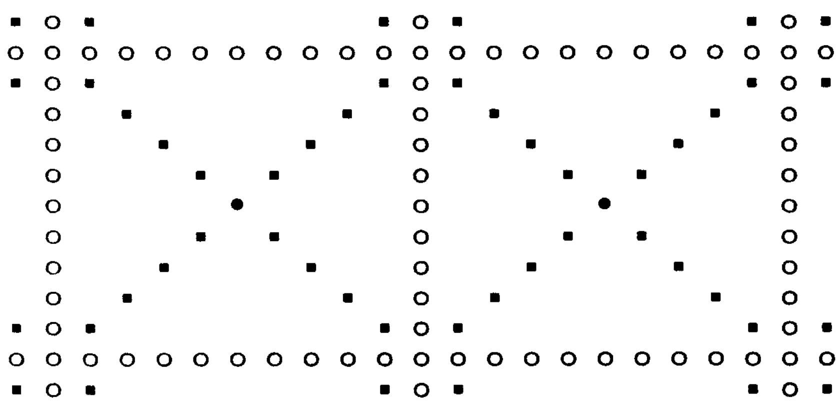
The arguments made in equations (78) through (81) can be repeated in a parallel fashion with the universal displacement current and universal displacement conductivity. One may view this application of Gauss' law as an integral formalism embodied by Kirchoff's rules for a universal displacement current. Each node within the mesh represents a small 'Gaussian' enclosure. The rules of displacement flux/current continuity are then applied to the boundaries between contacting nodes [37, 48, 49].

BOND OR SITE CENTERED PIXEL PERCOLATION APPROXIMATIONS

The generalized path admittance characteristics g_{ij} as given by equation (81) may be assigned on the basis of two pixel percolation approximations as illustrated in Figure 2: 1) site



a) Each path linking adjoining nodes in the site pixel percolation approximation straddles two pixel cell regions. Since a single constituent permittivity is assigned to each pixel, the admittance characteristic of the path is the series combination of the permittivities of both regions.



b) In the bond pixel percolation approximation each displacement flux/current path resides within a single pixel cell region. The path is assigned the admittance characteristic of the permittivity of the pixel cell.

Figure 2. Site (a) and bond (b) pixel percolation approximations. [o o o] designates the pixel cell boundary, [▪ ▪ ▪] outlines a possible displacement flux/current path, and a [•] denotes a node where paths meet.

---

centered pixels and 2) bond centered pixels. In both instances, simulations yield similar trends in the results with some background changes in the overall percolation.

1) The pixel permittivity/conductivity assignment method in the case of site centered pixels is to assign each mesh node region to a pixel. In the site representation the internodal paths have two consecutive admittances/conductances which are treated as a

series combination.

2) The permittivity/conductivity assignments for the case of bond centered pixels are that each internodal path represents a pixel characteristic. Since each pixel has been assigned only one of the constituent characteristics, each path is not a combination of properties.

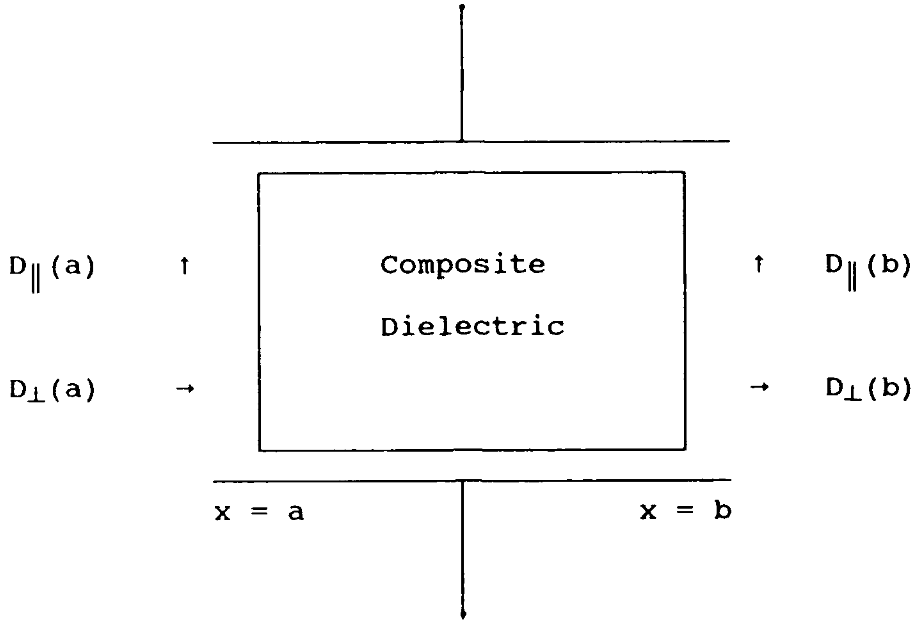
Both types of arrangements are discussed in the literature on percolation problems [37, 46, 47, 49].

LATERAL BOUNDARY CONDITIONS

The overall rectangular node mesh representing a dielectric composite specimen is solved with exciting electrodes or 'plates' placed in contact with the opposite faces. Continuity of the displacement field requires that the same amount of displacement flux/current which begins on one plate terminates on the other plate. This displacement flux/current boundary condition is satisfied at the plates when the exciting field is solved together with equation (81). This leaves an open question as to how to handle the displacement flux/current on the other (lateral) faces formed by the rectangular mesh that has been superimposed on the dielectric specimen. The displacement flux/current could still enter and exit at the other faces so long as the overall quantity is conserved. As illustrated in Figure 3, two types of lateral boundary conditions are implemented here.

The first and simplest condition is that of 'insulating' faces, wherein we consider the dielectric specimen to be electrically isolated or 'guarded' along the lateral boundaries. In this instance at the lateral boundaries, the normal component of the displacement field approaches zero as no displacement flux can leave the specimen. The 'insulating' boundary condition can be imposed exactly with the computer model although in physical reality this is more difficult to do because of the effect of fringing fields. This condition can be implemented through equation (81) by having the lateral mesh nodes interact only with adjacent interior mesh nodes. The 'insulating' boundary condition is applicable to isolated specimen samples or lattice cells with mirror symmetries and planes. For example, an ellipsoid has symmetry planes between each pair of semi-axes, and the solution for an octant of the ellipsoid is also the solution for a cubic repeating lattice of ellipsoids with the same repeating symmetry boundaries.

A second type of boundary condition is that of 'periodic' boundaries in which the specimen represents a repeated cell in a cyclic lattice structure. In this case the magnitude of the normal component of the displacement field is the same at similar locations on opposite lateral faces. In some cases either boundary condition can be used on the same problem.



Insulating

$D_{\parallel}(a) \neq D_{\parallel}(b) \neq 0$

$D_{\perp}(a) = D_{\perp}(b) = 0$

Periodic

$D_{\parallel}(a) = D_{\parallel}(b) \neq 0$

$D_{\perp}(a) = D_{\perp}(b) = 0$

Figure 3. Lateral boundary conditions on the composite dielectric. 'Insulating' means that no displacement flux/current enters or exits the lateral faces. 'Periodic' means that entering or escaping flux/current on a lateral face must enter at a similar location on the opposite lateral face such that overall displacement flux/current conservation is maintained.

In the case of random isotropic media the lateral displacement field becomes a vanishingly small statistical fluctuation and the choice between either set of boundary conditions has little effect on the overall solution. If the distribution of dielectric constituents is not uniformly random but shows an overall anisotropy, a lateral displacement flux/current may occur and the macroscopic permittivity will be nonuniformly tensored.

SOLUTION AND COMPUTER IMPLEMENTATION OF THE NETWORK

The mesh approximation is equivalent to an electrical network or a finite difference grid scheme. The nodes are the intersections between displacement flux/current paths. The electrical elements are the displacement flux/current path admittances between the nodes. The N nodes infer that there are N simultaneous node equations plus one for the exciting node. An important prerequisite to solving this set of N+1 simultaneous

equations is the choice of a numbering scheme. In the scheme adopted here, the nodes are numbered sequentially from the ground electrode and in relation to (x,y,z) coordinate location. The i^{th} node number is

$$i = x + (y-1)\ X_m + (z-1)\ X_m\ Y_m \tag{82}$$

where (x,y,z) are integers in the range $(1..X_m,\ 1..Y_m,\ 1..Z_m)$. The node address may be specified by the node number i or by the coordinates (x,y,z). The values X_m, Y_m, and Z_m are the extents of the rectangular mesh array in each direction.

The N+1 simultaneous node equations are solved directly and methodically by means of Gauss-Seidal elimination, a plodding but sure fire technique which can be adapted to the mathematics of complex numbers. Back substitution may be invoked to double check the consistency of a numerical solution. By solving the node equations in a sequence that roughly follows the order of the expected node potentials (i.e., the solution is worked from the ground potential up), one can minimize the buildup of truncation errors. The neighbor nodes of the boundary nodes wrap around to the opposite lateral face in the case of periodic or cyclic boundary conditions.

INTERACTION MATRIX

The set of N + 1 mesh equations as formed from equation (81) can be cast into a matrix form. The elements of this matrix are the interactions between nodes. The elements along the main diagonal of the matrix are self-interaction terms corresponding to the sum of all admittance paths connecting to a node. The other elements are the neighbor interactions. When the neighboring node directly adjoins, the corresponding matrix element is the negative admittance of the path. When the nodes are not directly joined by a single bond path, the matrix element is zero. The matrix formalism is convenient because many standard mathematics packages have libraries and function capabilities for matrix operations. The set of equations in matrix form may be written

$$\{\mathbf{\Phi}\} = \{\mathbf{g}\}\ \{\mathbf{V}\} \tag{83}$$

where $\{\mathbf{\Phi}\}$ is a column matrix whose elements are the total displacement flux at each node. According to the continuity requirement, all elements of $\{\mathbf{\Phi}\}$ are zero except that of the exciting flux. $\{\mathbf{V}\}$ is a column matrix whose elements are the node potentials and $\{\mathbf{g}\}$ is a matrix representing the admittance interactions between nodes. The admittance matrix $\{\mathbf{g}\}$ is symmetric and most off-diagonal elements are zero as they represent interactions between pairs of other than nearest neighbor nodes. (Classical rules of electromagnetism do not allow flux to jump from one region of space to another without an intervening path, thus interactions occur only between adjoining node pairs.) The quantum mechanical formalism does allow for

nonlocal jumping of displacement flux/current and hence more non-zero terms in the admittance matrix; however, in this report we will only deal with the classical case.

In the sparse symmetric matrix case examined here, a judicious Gauss-Seidal elimination progression proved to be substantially faster than full matrix inversion, especially when the N+1 equations become large. Equation (82) tells us that the maximum node number is the mesh box size which is $N+1 = (X_m\ Y_m\ Z_m)+1$. The number of non-zero terms in any row is only the number of neighbors in actual contact. By selecting a node numbering scheme as compact as possible between neighbors it is possible to solve the matrix with a minimum of memory space and computer operation. The solution time for the procedure implemented here went roughly as the square of the admittance interaction matrix size N+1. This allowed simulations of pixels and their mesh networks of up to about 40x40 in two dimensions and 12x12x12 in three dimensions.

The exponential averaging factor 'α' is computed by iteration of equation (1) until a consistent solution is reached between the resultant and constituent permittivities. The depolarization factor 'A' for binary composites can be solved from a quadratic solution of equation (2). The depolarization factor for composites having more than two constituents must be solved iteratively until a self-consistent solution of equation (2) is obtained.

VERIFICATION OF NUMERICAL SOLUTION

The numerical analysis has been implemented using the Hewlett-Packard HP Basic/UX 6.0 programming language, also known as Rocky Mountain Basic. This language follows the IEEE Std 754-1985 for binary numbers. The code is run on an HP 9000 Series 300 workstation with a machine precision of 8 bytes for real numbers and 16 bytes for complex numbers. The precision for complex numbers equates to a precision in the mantissa of about one part in 10^{15}. The accuracy of the numerical solution has been tested in several ways.

The first and most obvious test was the calculation of the macroscopic response of a sequence of layers with surface normals either aligned along or normal to the electric field direction. This could be accomplished by randomizing selectively on successively the $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes. No deviations from the expected limiting Wiener Bounds were found to occur other than truncation errors in the lowest few mantissa bits. A second test was to examine the stability of a given composite layout as it was enlarged or symmetrically folded into another pixel size. Since the physical problem is still the same in the symmetrical sense, then the expected solution cannot vary. The observed results were consistent with our expectations.

We had two methods of matrix inversion available in solving

the nodal analysis. One was the intrinsic matrix inversion built into the HP Basic language and the other was the tailored sparse-advantaged Gauss-Siedal elimination that we developed for the program. The two methods agreed with each other to within machine precision. Furthermore, in either mode the solutions could be checked by back substitution comparison to the original problem. We found errors no worse than in the lowest two or three mantissa bits.

RESULTS

The results of our work are displayed in Figures 4 through 31. Both the exponential averaging factor 'α' and the depolarization factor 'A' are displayed as the volume fraction of the constituents is changed. Both the exponential averaging factor 'α' and the depolarization factor 'A' are real numbers (as opposed to complex numbers) in causal systems, even when the permittivities of the constituents are complex. This fact provides an additional simplification of the results. In Figures 6 through 27, the real part of the averaging factor of interest is denoted with a (+) and the imaginary part with a (-) on the graphs. The imaginary part remains near zero (as expected) and tends to fluctuate markedly less than its real counterpart.

Figures 4 and 5 show respectively the resultant permittivity versus constituent volume for isotropic binary composites of $\epsilon_1=1$, $\epsilon_2=2$ and $\epsilon_1=1$, $\epsilon_2=10$. In each case the Wiener bounds are drawn along with several curves at equidistant spacings of exponential averaging and depolarization factors. For the case of nearly equal constituent permittivities as shown in Figure 4, the exponential averaging and depolarization curves are nearly indistinguishable. For larger permittivity differences as shown in Figure 5, the two formulae yield distinctly different results. In these figures, the permittivity is displayed directly to demonstrate that data can be presented in this way, but the curves are squeezed at the ends and lie within a banana-shaped envelope. With complex permittivities of real and lossy parts the data become even more difficult to display. Subsequent figures are thus presented by exponential averaging and depolarization factor windows whose abscissa limits are the Wiener bounds and whose ordinate limits are the minimum and maximum possible constituent volume fractions.

A number of cases are examined and displayed in terms of the exponential averaging and depolarization windows. The first case shown in Figures 6 and 7 is that of three-dimensional spheres spaced in an infinite cubic lattice. For small volume fraction of spheres, this case corresponds correctly to the far field limit of the depolarization expected for dielectric spheres embedded within a host. However, as the volume fraction of the spheres increases, near field changes occur and the depolarization departs from the far field approximation.

Figures 8 through 27 present various Monte Carlo simulations both in two and three dimensions. The two-dimensional cases in Figures 8 through 13 are complex valued extensions of real-valued analyses from the earlier work [50]. The numerical permittivity simulations have been carried out with both bond and site centered pixel arrangements.

There generally can be a multitude of constituent grain shapes for various heterogeneous composites or mixtures that can be modeled. The simplest possibility is a random shuffling of iso-sized constituent grains. The iso-sized grain case would probably best correspond with molecular mixing without the chemical and quantum interactions. In geology, 'sandy' composites or iso-sized conglomerates could be examples. The iso-sized constituent grain model is mainly what has been approximated by our network models in Figures 8 through 23. Another likely possibility is a composite in which there is a range of constituent grain sizes randomly shuffled. A 'marbled' mixture would have some distribution of assorted constituent grain sizes. Many natural composites exhibit this feature to some degree. Figures 24 through 27 show network simulations where a preliminary effort has been made to include a limited range of constituent grain sizes. Our results indicate an increased scatter which is due to the additional fluctuations induced by having the larger grains present. Also our results still closely approximate the iso-sized case in that the effective medium or depolarization factor 'A' is mostly constant. The calculated values of Lichtenecker's factor 'α' would remain more constant if the same self-similarity pattern were maintained over a composite sample which is much larger than the largest constituent grain. Obviously our network simulations are very limited in this respect. Only near the percolation threshold transition is the self-similarity aspect primarily evident in the grain layout of the iso-sized shuffling case [38]. In the even-numbered Figures 8 through 26 the percolation is approximately marked at the centers of transition in the 'α'-factor.

Indications from the numerical simulations are that both the exponential averaging factor 'α' and the depolarization factor 'A' are nearly congruent for representing the resultant permittivity of composite mixtures where the constituent permittivities differ by less than a factor of two such as in Figures 14 and 15. When the constituent permittivities differ by more than a factor of two, the exponential averaging factor and depolarization factor diverge according to the mixture type. This is evident in the Figures 8 through 13 and 16 through 27. The implication is that the resultant permittivity is dependent on the randomness type involved in the composite, such as whether the constituent grains are iso-sized or have a self-similar range of assorted sizes.

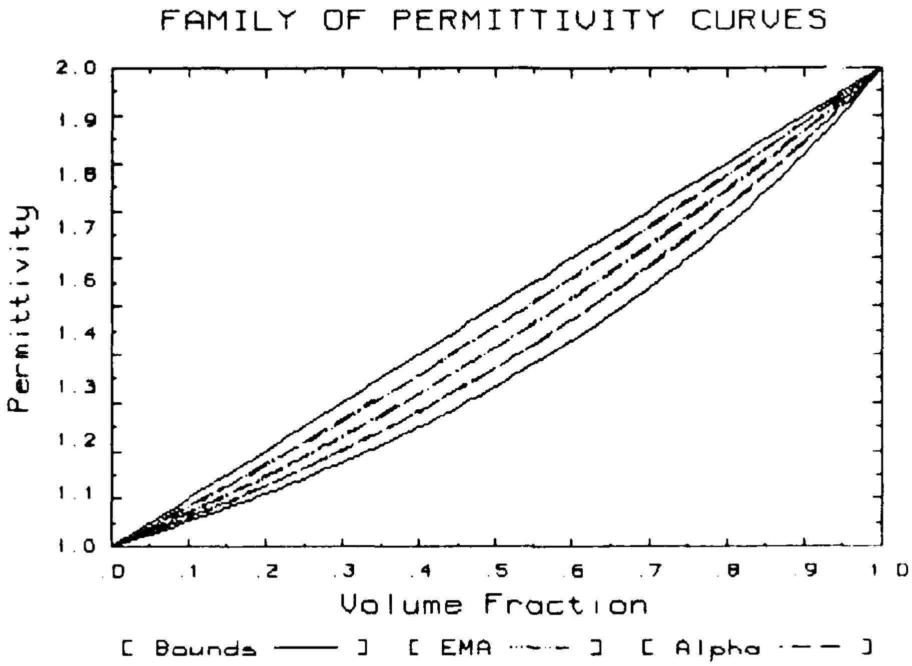


Figure 4. Permittivity curves at ratio $\epsilon_2/\epsilon_1=10$. The curves situated between the Wiener bounds are equally spaced in α and A ranges respectively. ($\alpha=-.50$, A=.75, $\alpha=.0$, A=.50, $\alpha=.50$, A=.25)

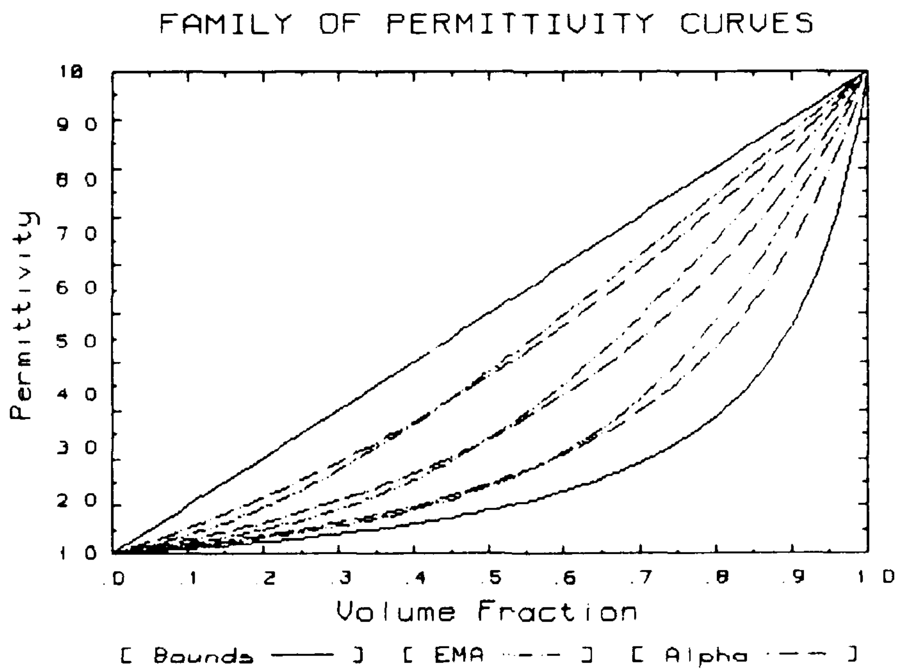


Figure 5. Permittivity curves at ratio $\epsilon_2/\epsilon_1=100$. As above the curves are equally spaced. Note that the bounds envelope expands and each curve pair is less coincident for the larger ratio.

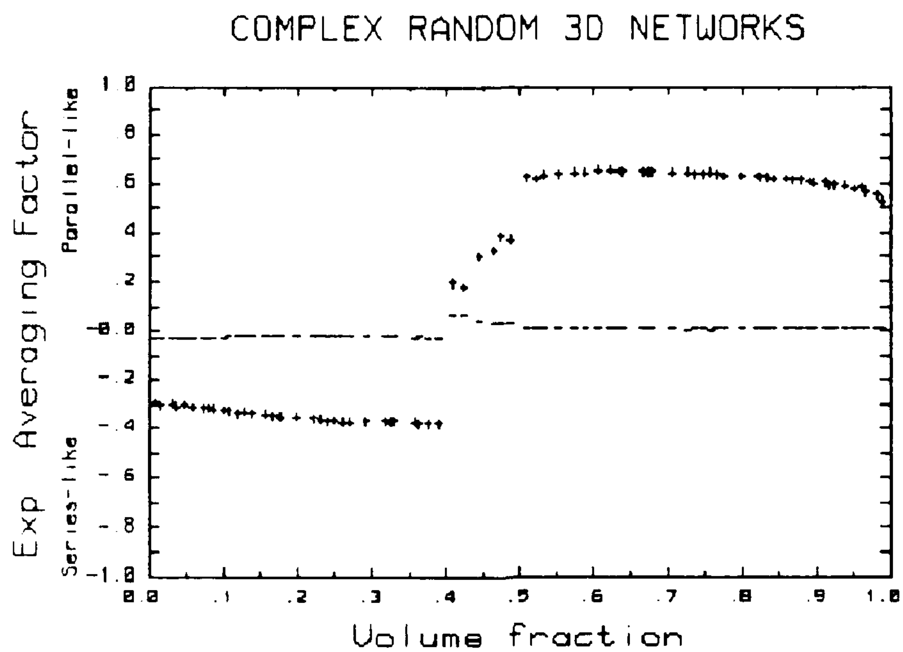


Figure 6. Cubic l-ttice of spheres with permittivity (1,1000) embedded within a host of permittivity (1,0), α-windowed. Note the percolatic. ·ump which occurs when spheres make contact at closest packing. Resolution 20x20x20 per cell.

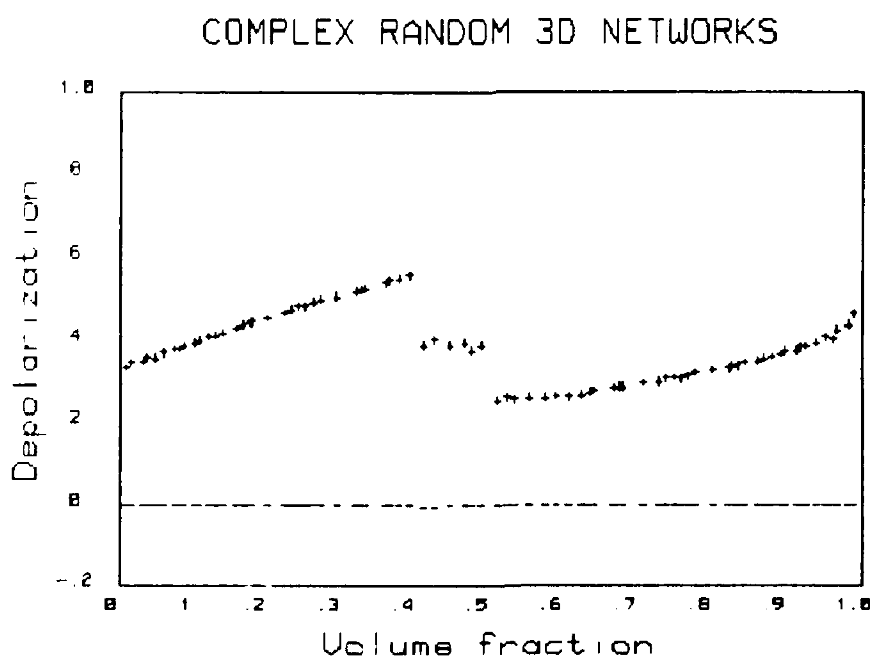


Figure 7. Cubic lattice of spheres with permittivity (1,1000) embedded within a host of permittivity (1,0), A-windowed. The depolarization matches the expected far field limit values at the ends of the volume packing range.

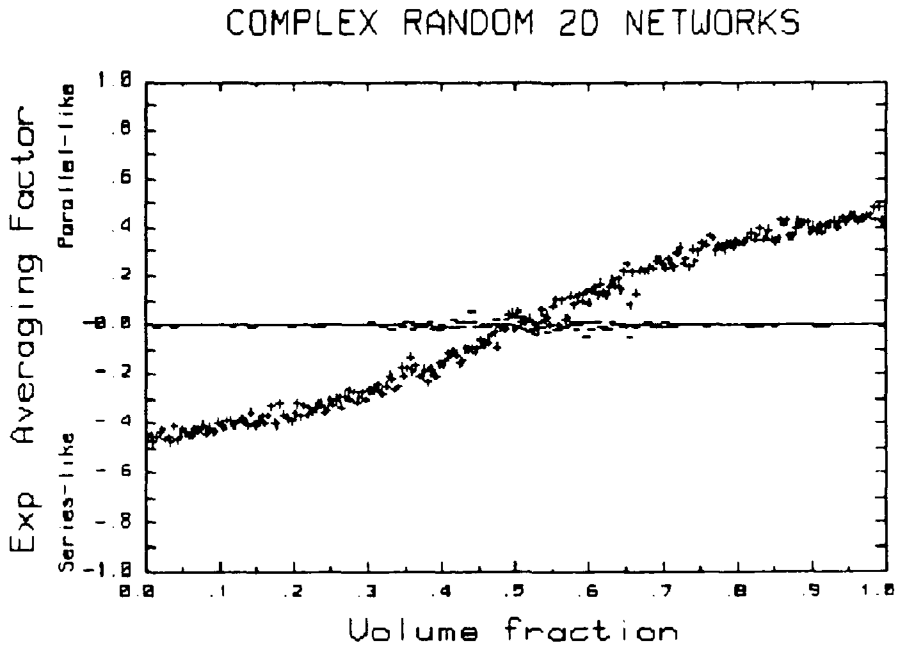


Figure 8. Two-dimensional composite mixture, α-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(0,100)$. Each bond or connecting path is randomly assigned one of the permittivities.

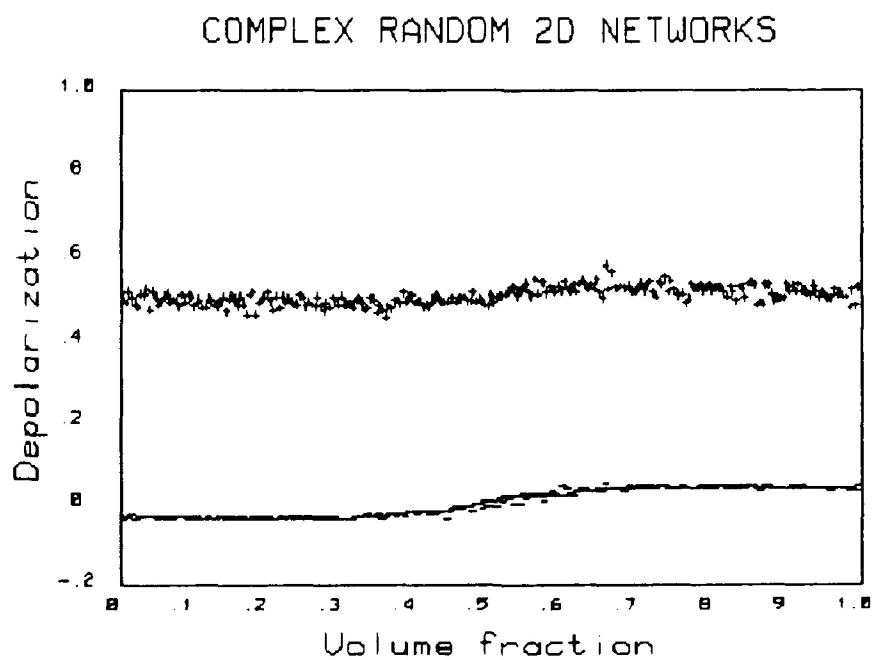


Figure 9. Two-dimensional composite mixture, A-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(0,100)$. Each bond or connecting path is randomly assigned one of the permittivities.

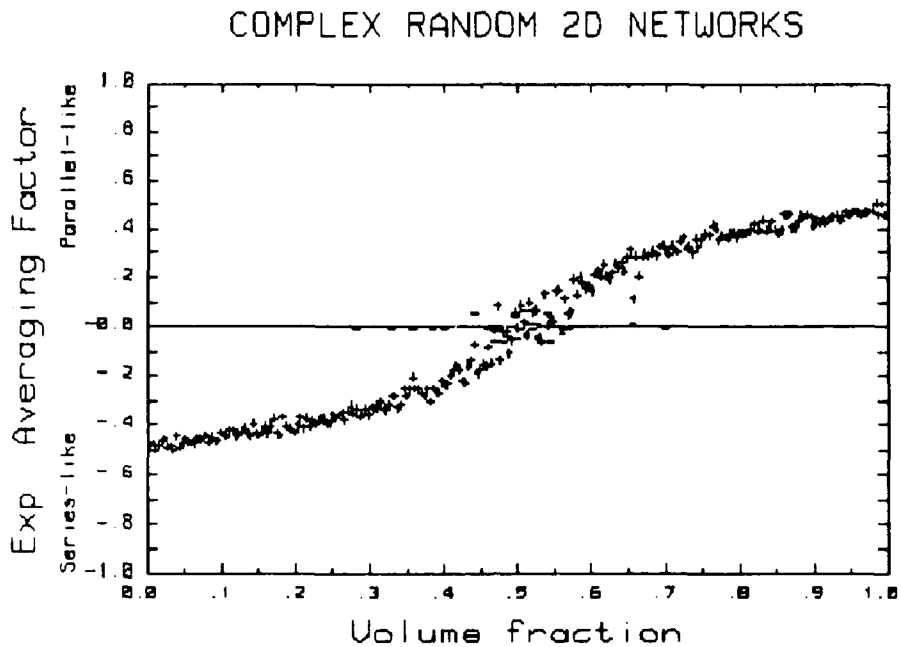


Figure 10. Two-dimensional composite mixture, α-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(0,1000)$. Each bond or connecting path is randomly assigned one of the permittivities. Resolution size at 40x40 bonds.

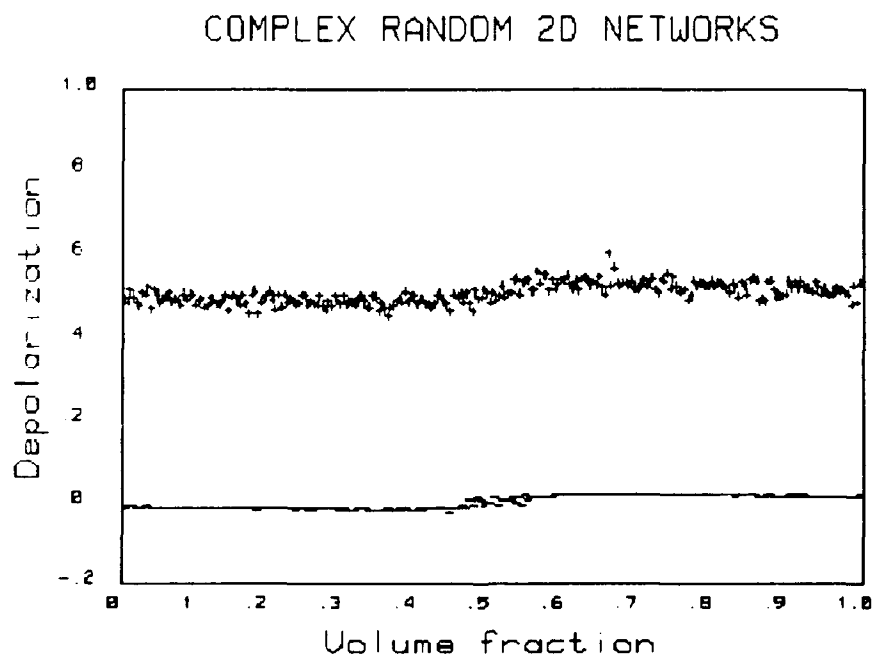


Figure 11. Two-dimensional composite mixture, A-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(0,1000)$. Each bond or connecting path is randomly assigned one of the permittivities.

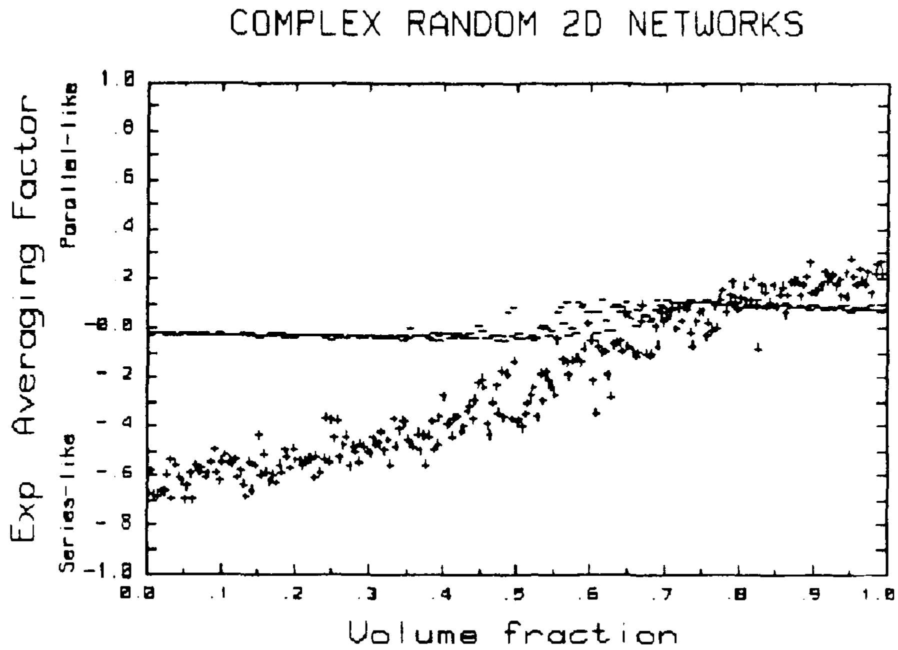


Figure 12. Two-dimensional composite mixture with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(0,100)$, α-windowed. Each site or path cluster surrounding a node is randomly assigned one of the permittivities. Resolution or network size is 20x20 sites.

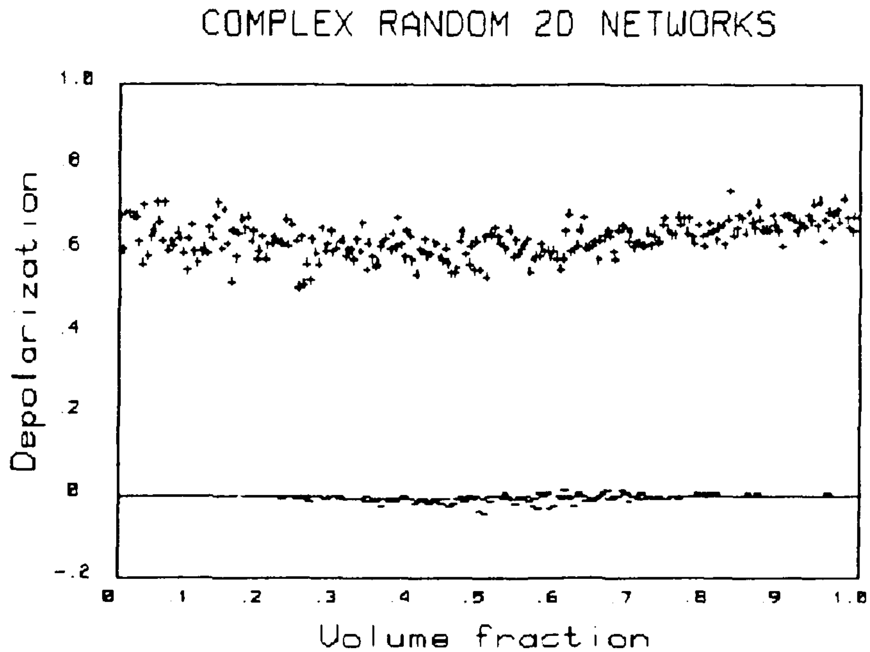


Figure 13. Two-dimensional composite mixture with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(0,100)$, A-windowed. Each bond or node cluster is randomly assigned one of the permittivities.

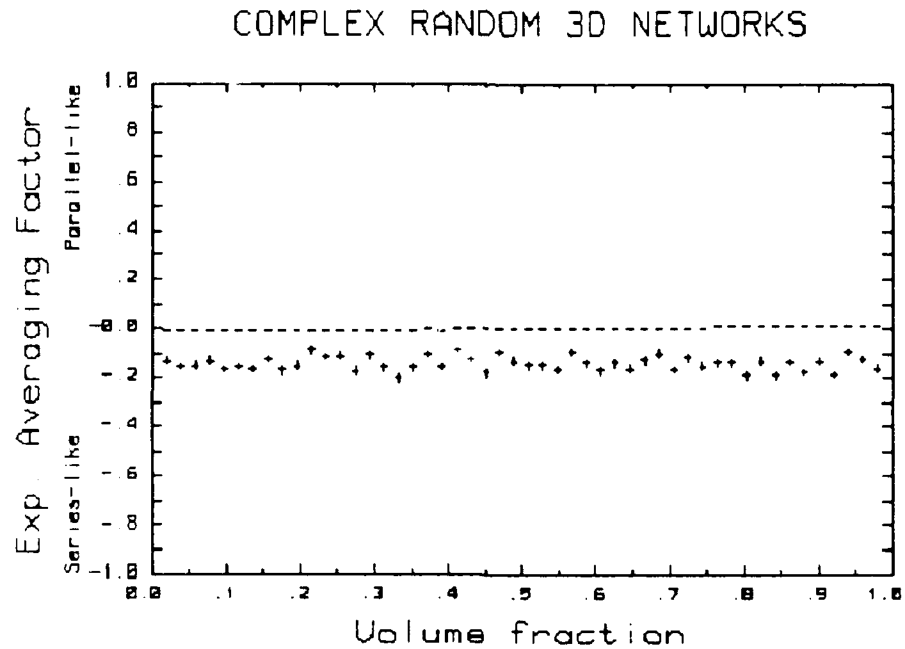


Figure 14. Three-dimensional composite mixture with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(1,1)$, α-windowed. Each site or node cluster is randomly assigned one of the permittivities. Resolution size at 10x10x10 bonds.

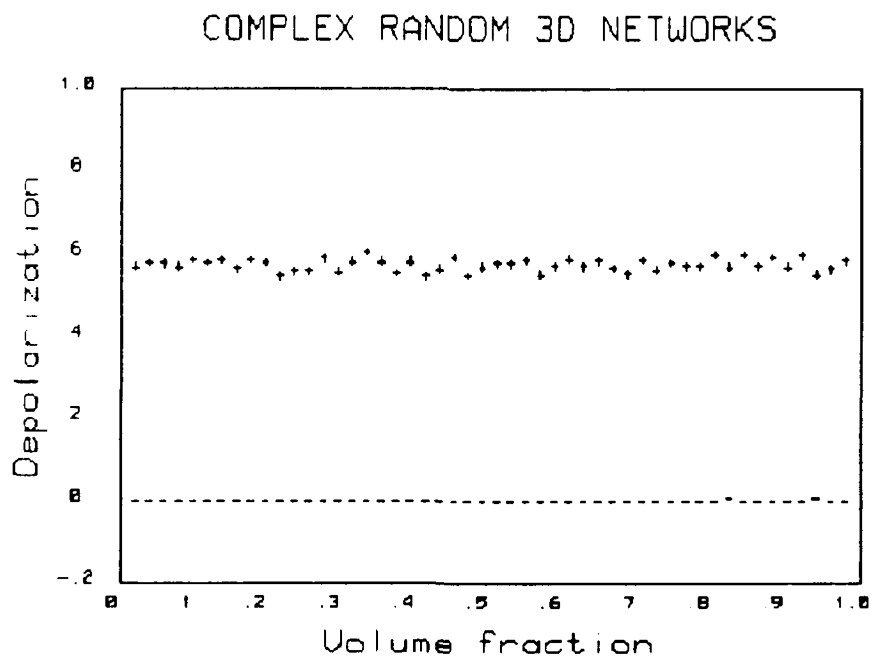


Figure 15. Three-dimensional composite mixture with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(1,1)$, A-windowed. Both 'α' and 'A' curves are nearly coincident in permittivity between the Wiener Bounds.

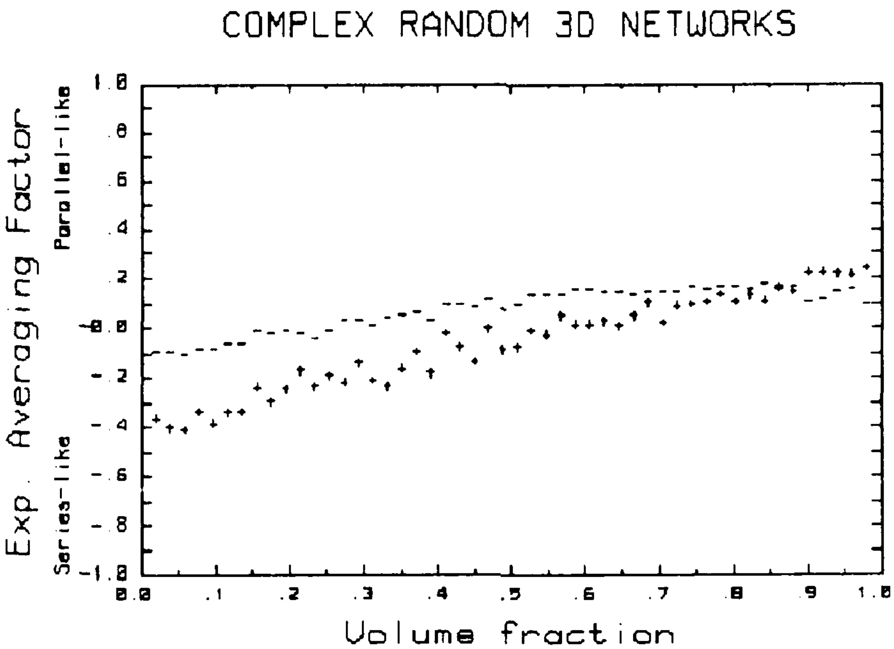


Figure 16. Three-dimensional composite mixture, α-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(1,10)$ randomly shuffled in a site grid, 10x10x10. Ratio of ϵ_2/ϵ_1 is $(1,10)/(1,0)=(1,10)$.

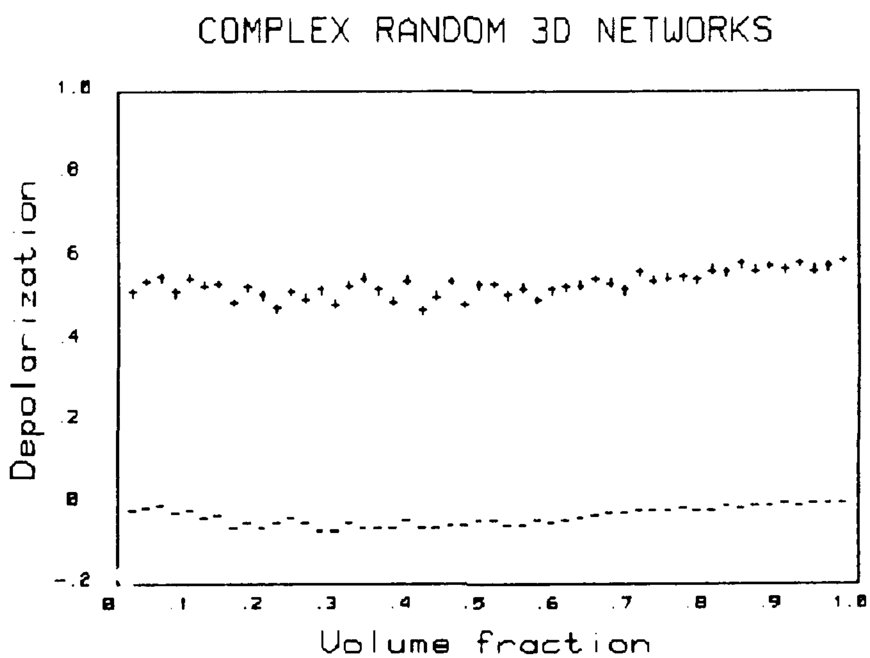


Figure 17. Three-dimensional composite mixture, A-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(1,10)$ randomly shuffled in a site grid, 10x10x10. Insulated boundary conditions apply. The depolarization remains at about .5.

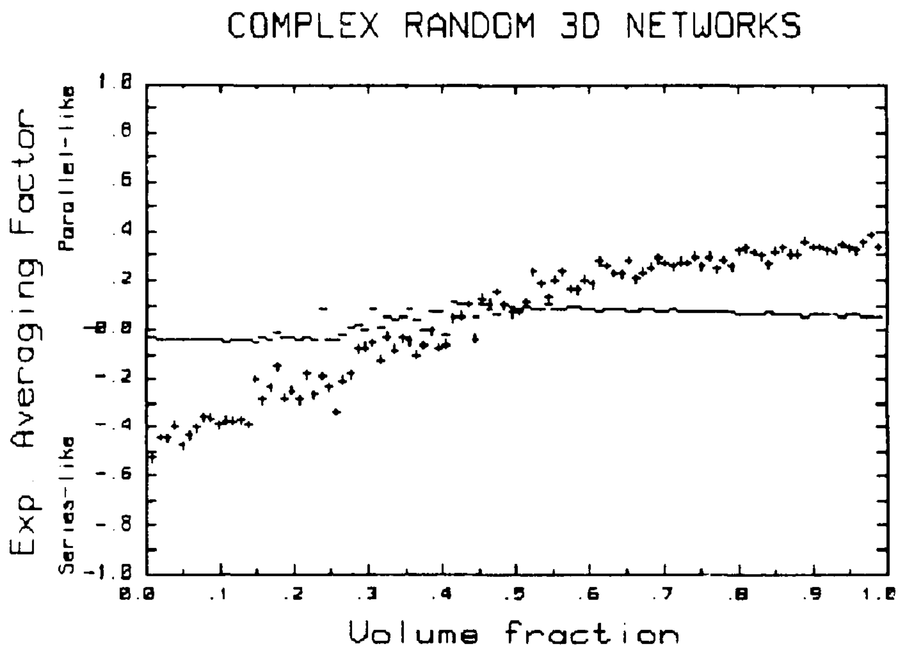


Figure 18. Three-dimensional composite mixture, α-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(1,100)$ randomly shuffled in a site grid, 10x10x10.

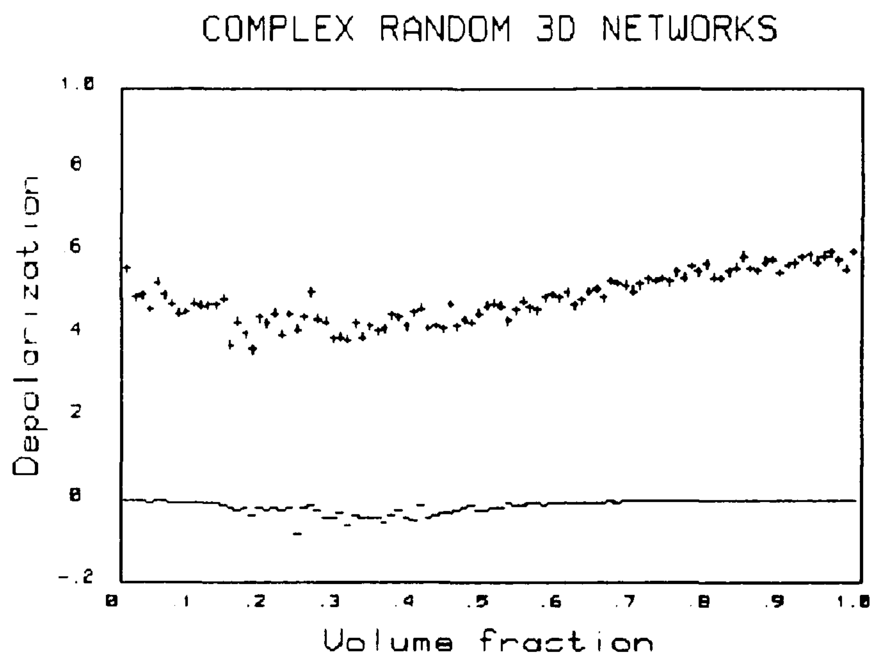


Figure 19. Three-dimensional composite mixture, A-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(1,100)$ randomly shuffled in a site grid, 10x10x10. The depolarization slightly deforms with a minimum around the percolation threshold.

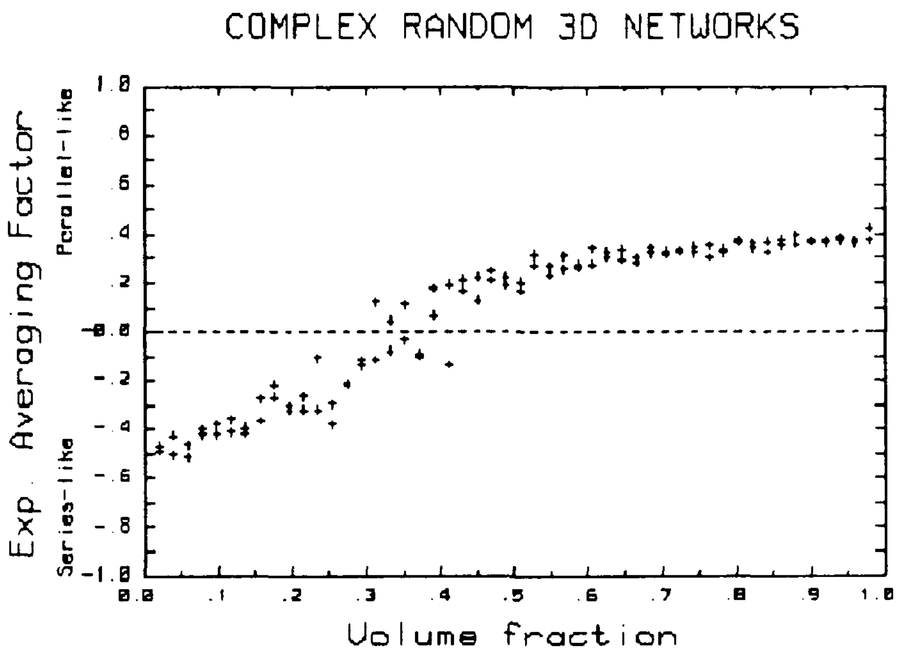


Figure 20. Three-dimensional composite mixture, α-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(1,1000)$ randomly shuffled in a site grid, 10x10x10. Insulated lateral boundary conditions apply.

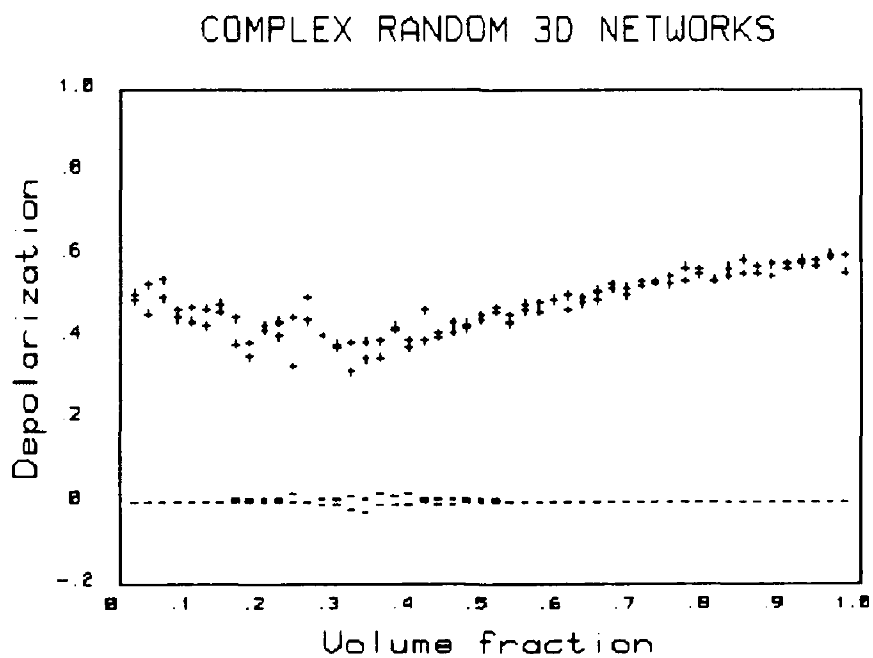


Figure 21. Three-dimensional composite mixture, A-windowed, with constituent permittivities of $\epsilon_1=(1,0)$ and $\epsilon_2=(1,1000)$ randomly shuffled in a site grid, 10x10x10. The depolarization deforms with a minimum around the percolation threshold.

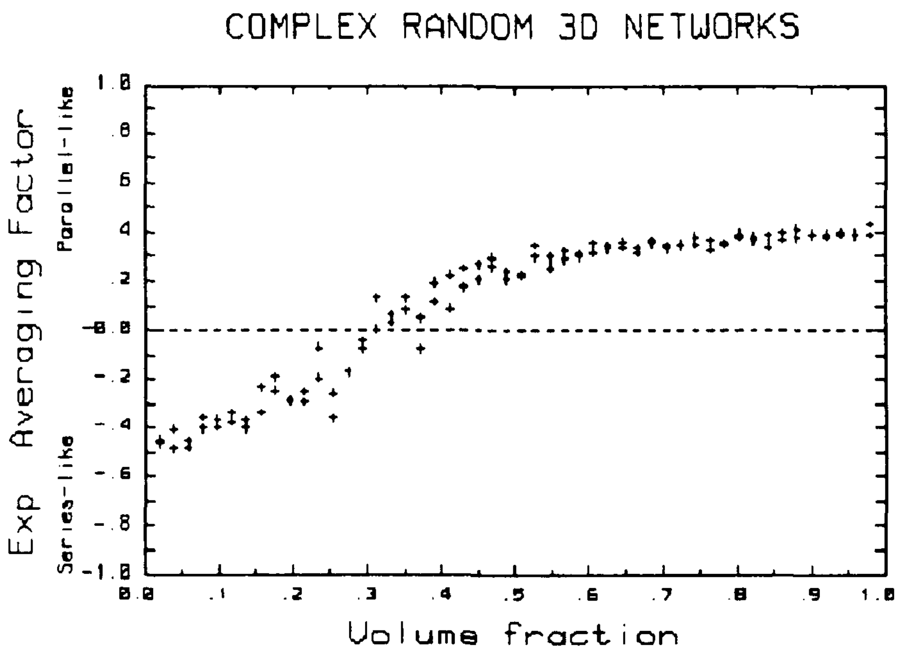


Figure 22. Three-dimensional composite mixture, α-windowed, with constituent permittivities of ϵ_1=(1,0) and ϵ_2=(1,1000) randomly shuffled in a site grid, 10x10x10, with periodic boundary conditions.

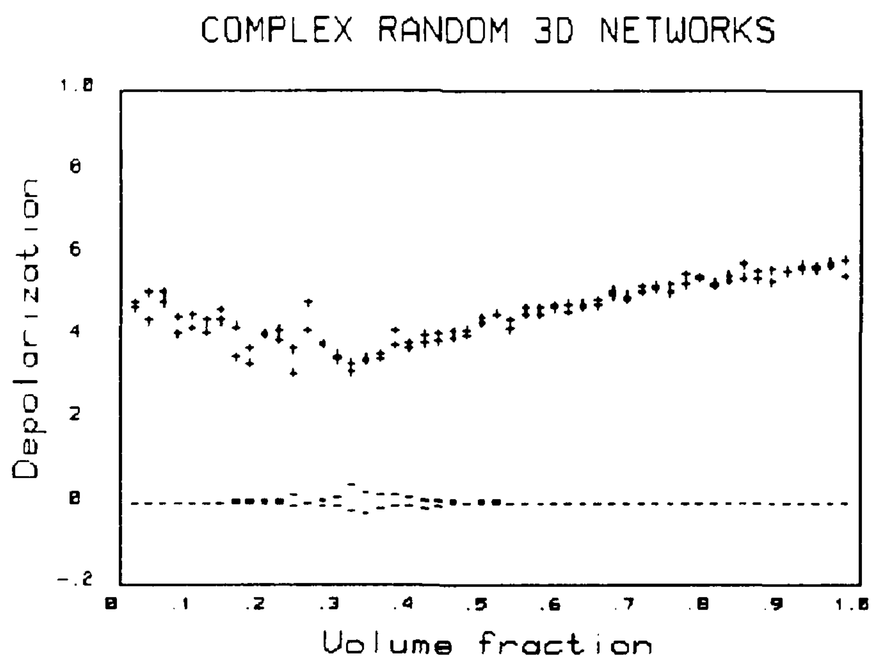


Figure 23. Three-dimensional composite mixture, A-windowed, with constituent permittivities of ϵ_1=(1,0) and ϵ_2=(1,1000) randomly shuffled in a site grid, 10x10x10, with periodic boundary conditions. The change from insulated to periodic boundary conditions has little effect on this random case.

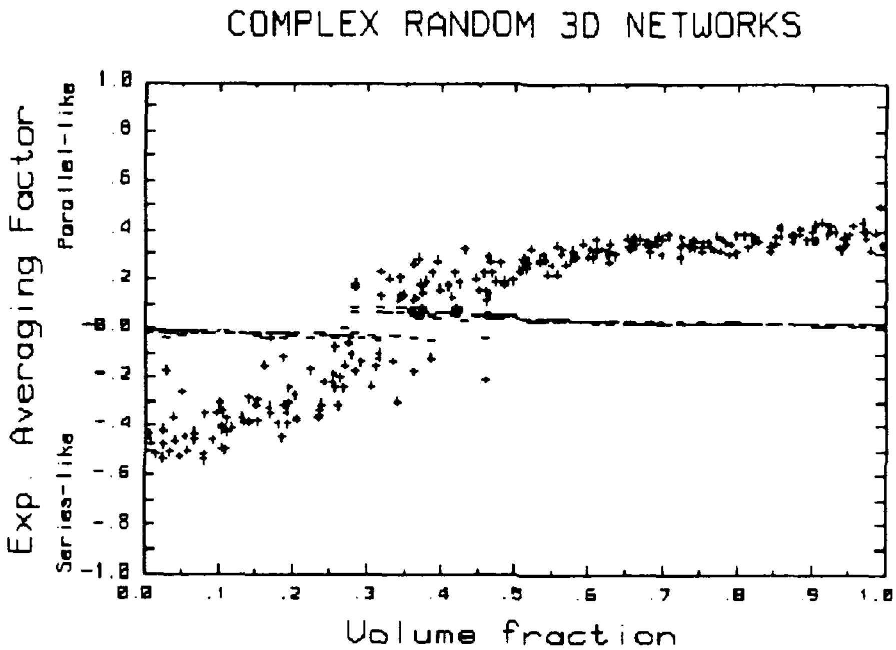


Figure 24. Three-dimensional composite simulation with 8x8x8 sites, α-windowed, with permittivities $\epsilon_1=(1,0)$ and $\epsilon_2=(0,1000)$. Each size stage 2x2x2, 4x4x4, and 8x8x8 is randomly scaled and shuffled.

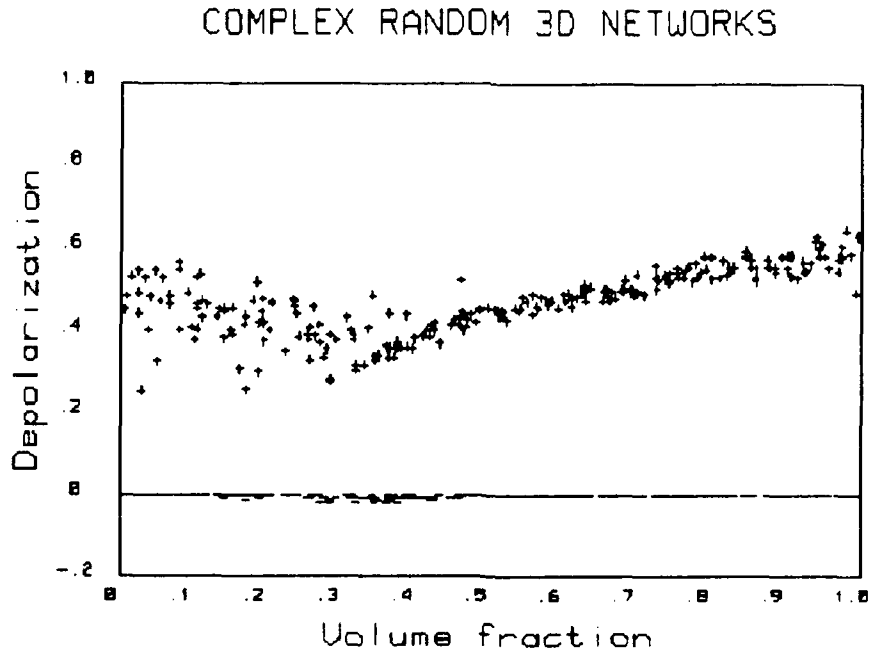


Figure 25. Three-dimensional composite simulation with 8x8x8 sites, A-windowed, with permittivities $\epsilon_1=(1,0)$ and $\epsilon_2=(0,1000)$. Each size stage 2x2x2, 4x4x4, and 8x8x8 is randomly scaled and shuffled. Note that the random composites grains have more scatter as a result of the wider size distribution.

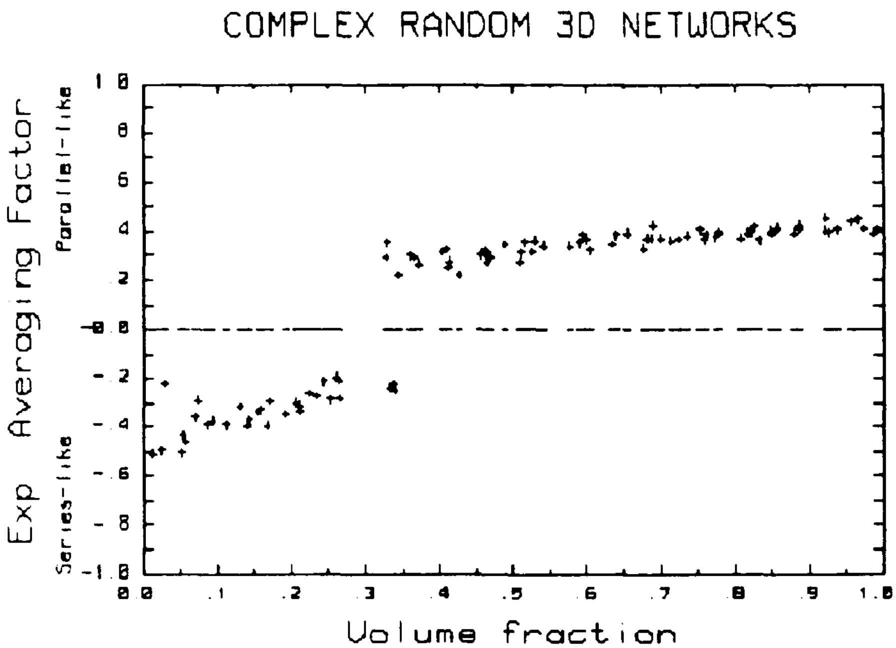


Figure 26. Three-dimensional composite simulation with 8x8x8 sites, α-windowed, with permittivities $\epsilon_1=(1,0)$ and $\epsilon_2=(0,10^9)$. Each size stage 2x2x2, 4x4x4, and 8x8x8 is randomly scaled and shuffled.

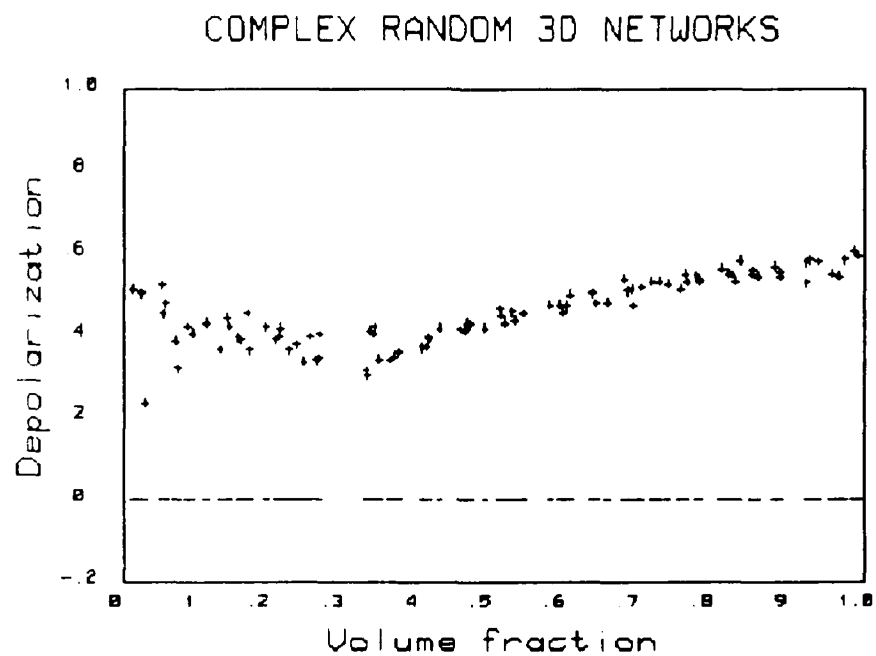


Figure 27. Three-dimensional composite simulation with 8x8x8 sites, α-windowed, with permittivities $\epsilon_1=(1,0)$ and $\epsilon_2=(0,10^9)$. Each size stage 2x2x2, 4x4x4, and 8x8x8 is randomly scaled and shuffled. Even at this large ratio of permittivities, back substitution revealed little error.

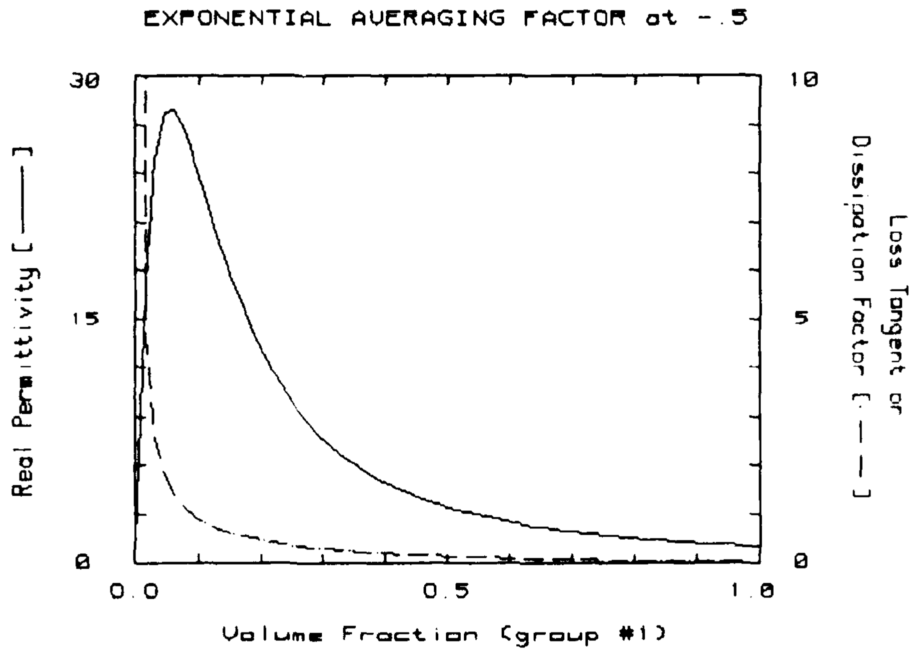


Figure 28. The Maxwell-Wagner effect of mixing a low loss constituent with a high loss constituent for permittivities $\epsilon_1=(1,0)$ and $\epsilon_2=(0,100)$ and $\alpha=-0.5$.

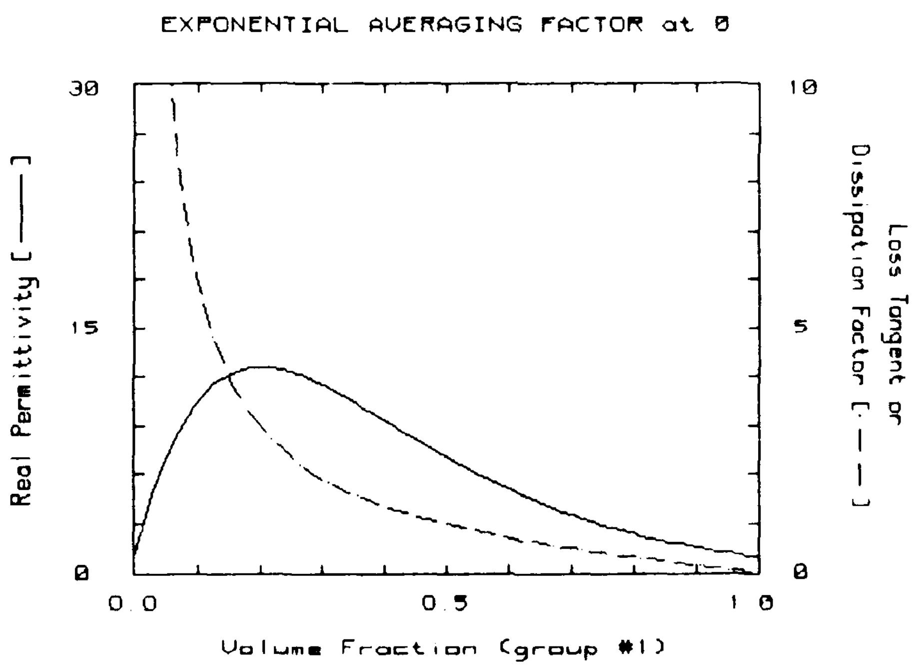


Figure 29. The Maxwell-Wagner effect of mixing a low loss constituent with a high loss constituent for permittivities $\epsilon_i=(1,0)$ and $\epsilon_2=(0,100)$ and $\alpha=0$.

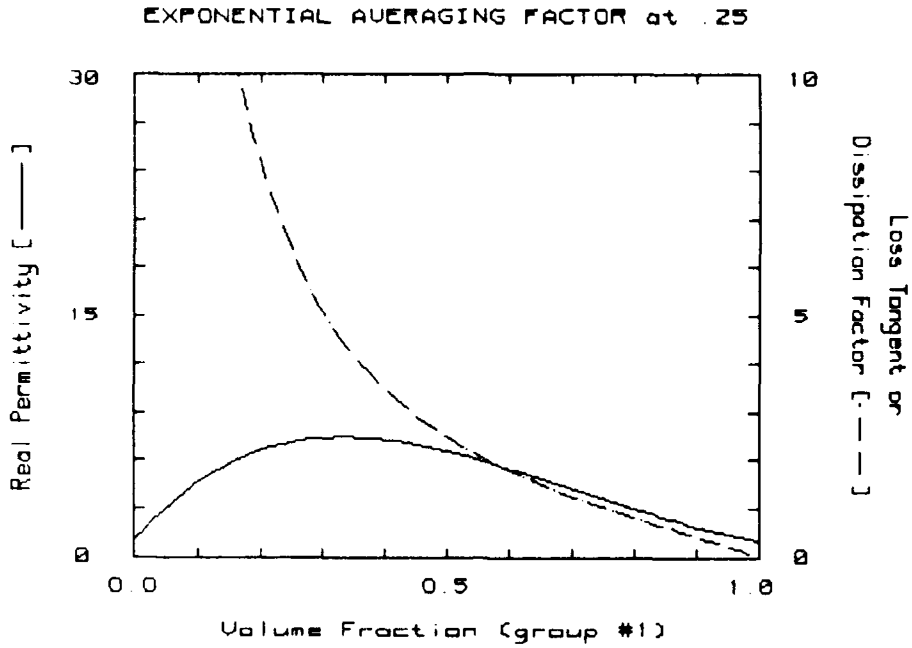


Figure 30. The Maxwell-Wagner effect of mixing a low loss constituent with a high loss constituent for permittivities $\epsilon_1=(1,0)$ and $\epsilon_2=(0,100)$ and $\alpha=0.25$.

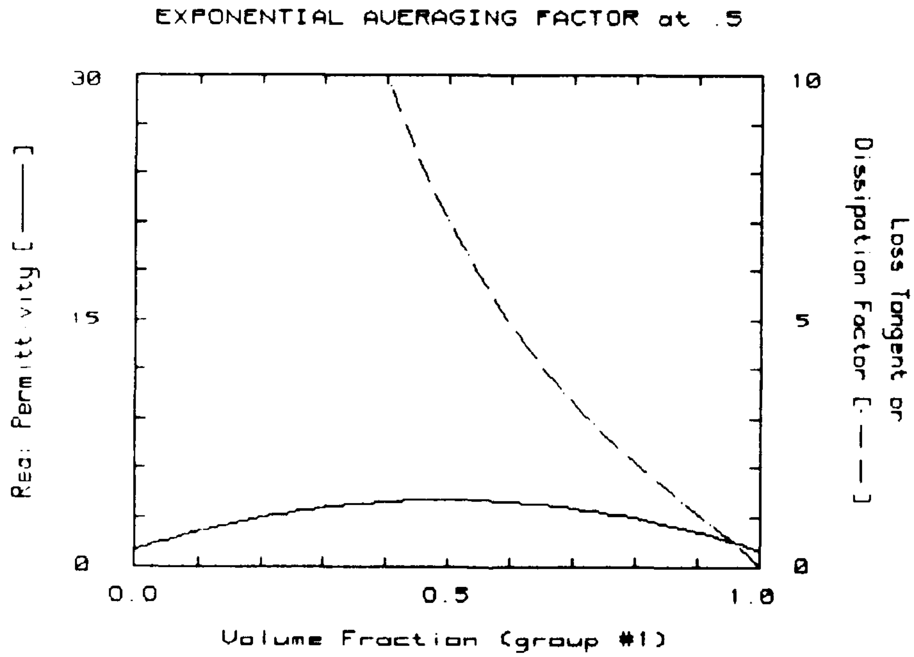


Figure 31. The Maxwell-Wagner effect of mixing a low loss constituent with a high loss constituent for permittivities $\epsilon_1=(1,0)$ and $\epsilon_2=(0,100)$ and $\alpha=0.5$.

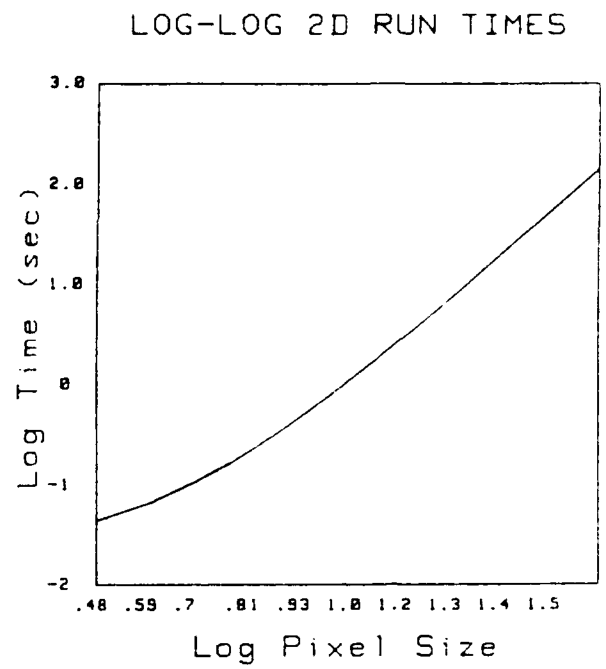


Figure 32. Run times for two-dimensional simulations.

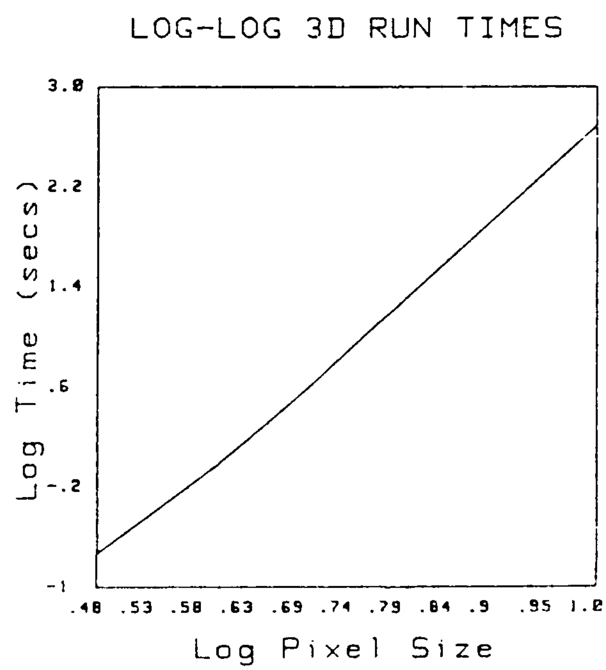


Figure 33. Run times for three-dimensional simulations.

FUTURE WORK

The completion of the design and computer coding for the bond and site models for composite dielectrics opens up a large range of research possibilities. Future work should include:

a) Analysis of composites with three or more components. Studies allowing mixture formulae to be employed within the pixels in a fashion self-consistent with the overall composite solution.

b) Studies of the mapping of the displacement field and the electric field within specimens.

c) Extension to cases of dyadic or tensorial permittivity constituents and composite dielectric response.

d) Study of fractal and self-similar composites especially random fractals and scaling of percolation. Fractal dimensionality and spatial correlation layout measurements would reflect the degree of self-similarity and also allow spectral dimensionality considerations.

e) Investigation of the Maxwell-Wagner effect over different types of composite mixtures.

f) Development of engineering design code, tables, and approximations which make these composite mixture formulae easily accessible.

g) Extension to pole-zero analysis and design of artificial dielectrics as well as the utilization of the Kramers-Kronig relations for cross-checking laboratory data as to causality or the interpolation and smoothing of partial data gaps. Examination of the relationship between our network studies and the phenomena of universal type of frequency dependence in ponderable dielectric properties as exhibited by many substances.

h) Studies of stretched, elongated, and stratified composites.

i) Studies of quantum overlap, quantum sensitive, and non-local effects.

REFERENCES

[1] T.R. Jow, and P.J. Cygen, "Dielectric Properties of Isopropyl Biphenyl and Propylene Carbonate Mixtures", CEIDP Conference Record, 1991.

[2] James Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., Vol. II, Art. 314-334, pp. 440-464, Clarendon Press, Oxford, 1892; also Dover Pub., 1952.

[3] Karl Willy Wagner, "Erklärung der dielektrischen Nachwirkungsvorgänge auf Grund Maxwellscher Vorstellungen (The After-effect in Dielectrics)", Archiv. f. Elektrotechnik, 2, pp. 371-387, 1914.

[4] Yasuo Kita, "Dielectric relaxation in distributed dielectric layers", J. Appl. Phys., 55, pp. 3747-3755, 1984.

[5] Karl von Lichtenecker und Karl Rother, "Die Herleitung des logarithmischen Mischungsgetzes aus allgeneinen Prinzipien der stationären Strömung", Physikalische Zeitschrift, Vol. 32, pp. 255-260, 1931.

[6] P.S. Neelakantaswamy, B.V.R. Chowdari, and A. Rajartnam, "Stochastic mixture model for powder dielectrics", J. Phys. D; Appl. Phys. (UK), pp. 1785-1799, 1983.

[7] Otto Wiener, "Die Theorie Des Mischkörpers Das Feld Der Stationären Strömung", Abhandlungen der Math.-Phys. Klasseder Keonigl. Sach. Ges. der Wiss., 32, Leipzig, pp. 509-600, 1912.

[8] Artur Büchner, "Das Mischkörperproblem in der Kondensatorentechnik", Wiss. Verofental. Siemens Werken, 18, 84, p. 204, 1939.

[9] D.A.G. Bruggeman, "Uber die Geltungsbereiche und die Konstanwerte der verschiedene Mischkörperformela Lichtenecker's", Physikalische Zeitschrift, 37, p. 906, 1936.

[10] Z. Hashin, and S. Shtrikman, "A Variational Approch to the Theory of the Effective Magnetic Permeability of Multiphase Materials", J. Appl. Phys., 33, p. 3125, 1962.

[11] Wayne R. Tinga, Multiphase Dielectric Theory - Applied to Cellulose Mixtures, Thesis, Dept. of Elec. Eng., Univ. of Alberta, Edmonton, Chap. 1, 1969.

[12] Ion Bunget and Mihai Popescu, Physics of Solid Dielectrics, Materials Science Monographs, 19, Elsevier Sci. Pub., Amsterdam, 1984.

[13] C.J.F. Böttcher, "The Dielectric Constant of Crystalline Powders", Rec. Trav. chim., renamed Rec. J. Roy. Neth. Chem. Soc., 64, 1945.

[14] D.A.G. Bruggeman, "Berechnung verschiedenier physikalischen Konstanten von heterogenen Substanzen", Annalen der physik, 24, pp. 636-679, 1935.

[15] J.E. Gubernatis, "Scattering Theory and Effective Medium Approximation to Heterogeneous Materials", AIP Conf. Proc. No. 40 on Electrical Transport and Optical Properties of Inhomogeneous Media, eds., J.C. Garland and D.B. Turner, A.I.P. Pub., New York, p. 84, 1978.

[16] C.J.F. Böttcher, Theory of Electric Polarization, "An ellipsoidal body in a dielectric", Elsevier Pub., Amsterdam, sec. 9, pp. 79-85, 1952.

[17] Julius A. Stratton, "Problem of the Ellipsoid", Electromagnetic Theory, McGraw-Hill, New York, Chap III, pp. 207-217, 1941.

[18] S. Wallin, Dielectric Properties of Heterogeneous Media, Thesis, Dept. of Physics & Astro., Univ. of Wyoming, Laramie, Wyoming, 1985.

[19] A.B. Baden Fuller, Microwaves, 2nd ed., Pergamon, Oxford, Chap. 6, pp. 148-149, 1979.

[20] K.G. Budden, The propagation of radio waves, Cambridge Univ. Press, Cambridge, Chap. 2, pp. 22-37, 1985.

[21] James Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., Vol. II, Chaps. IX & X or secs. 600-615, (see especially secs. 608-609, pp. 252-3), Clarendon Press, Oxford, 1892; also Dover Pub., 1952.

[22] Nicholas Bottka, "Dielectric Properties", Encyclopedia of Physics, 2nd ed., Rita G. Lerner and George L. Trigg, editors, VCH Pub., New York, pp. 249-251, 1991.

[23] John David Jackson, Classical Electrodynamics, 2nd ed., John Wiley, New York, Chap. 6-7, pp. 209-333, 1975.

[24] R.W. Sillars, "The Properties of a Dielectric Containing Semi-conducting Particles of Various Shapes", J. Instn. Elect. Engrs. (UK), 80, pp. 378-392, 1937.

[25] Arthur B. von Hippel, Dielectrics and Waves, M.I.T. Press, Cambridge, Mass., pp. 230-1, 1954.

[26] H.A. Kramers, "La diffusion de la lumiere par les atomes", Atti. Congr. Internat. dei Fisci Como 2, p. 545, 1927; also Collected Scientific Papers (of H.A. Kramers), North-Holland, Amsterdam, p. 333, 1956.

[27] R. de L. Kronig, "On the Theory of Dispersion of X-Rays",

J. Opt. Soc. Am., 12, p. 547, 1926.

[28] Richard L. Weaver and Yih-Hsing Pao, "Dispersion relations for linear wave propagation in homogeneous and inhomogeneous media", J. Math. Phys., 22(9), pp. 1909-1918, Sept. 1981.

[29] B.M. Tareev, Physics of Dielectric Materials, Eng. trans. by A. Troitsky, MIR Publishers, Moscow, Chapter 2, Sec. 2-6, pp. 116-125, 1979.

[30] J.A. Reynolds and J. M. Hough, "Formulae for Dielectric Constant of Mixtures", Proc. Phys. Soc. of Lon. B, 70, pp. 769-775, 1957.

[31] L.K.H. Van Beek, "Dielectric Behaviour of Heterogeneous Systems", Progress in Dielectrics, vol. 7, pp. 67-144, London:Heywood Books, 1961.

[32] Wayne R. Tinga and W. A. G. Voss, "Generalized approach to multiphase dielectric mixture theory", J. Appl. Phys., 44, p. 3897f, 1973.

[33] D.K. Hale, "The physical properties of composite materials", J. Mat. Sci., 11, p. 2105, 1976.

[34] Constantino Grosse and Jean-Louis Greffe, "Permittivité Statique Des Émulsions", J. de Chimie Physique et Phys. Chim. Bio., 76(4), pp. 305-327, 1979.

[35] P.N. Sen, C. Scala, and M. H. Cohen, "A self similar model for sedimentary rocks with application to the dielectric constant of fused glass beads", Geophys., 46, p. 781, 1981.

[36] R. Botet, and R. Jullien, "FRACTAL AGGREGRATES OF PARTICLES", Phase Transitions, Vol. 24-26, Gordon & Breach (UK), pp. 691-736, 1990.

[37] J.P. Clerc, et al, "The electrical conductivity of binary disordered systems, percolation clusters, fractals, and related models", Advances in Physics (UK), 39(3), pp. 191-309, 1990.

[38] Manfred R. Schroeder, Fractals, Chaos, Power Laws, ISBN 0-7167-2136-8, W.H. Freeman, 1990.

[39] V.I. Odelevski, "RASCHET OBOBSHCHENOY PROVODIMOCTI HETEROGENIE SISTEM", ZHURNAL TEKHNICHESKOY FIZIKI (Russian), 21(6), pp. 678-685, 1951; 21(11), pp. 1379-1382, 1951.

[40] Rolf Landauer, "The Electrical Resistance of Binary Metallic Mixtures", J. Appl. Phys., 23, p. 779, 1952.

[41] W.E.A. Davies, "The theory of composite dielectrics", J. Phys. D: Appl. Phys.(UK), 4, pp. 318-328, 1971.

[42] G.A. Niklasson, C.G. Granqvist, and O. Hunderi, "Effective medium models for the optical properties of inhomogeneous materials", Appl. Optics, 20(1), pp. 26-30, 1991.

[43] W. Lamb, D. M. Wood, and N. W. Ashcroft, "Long-wavelength electromagnetic propagation in heterogeneous media", Phys. Rev. B, 21, p. 2248, 1980.

[44] J.C. Maxwell Garnett, "Colours in Metal Glasses and in Metallic Films", Philo. Trans. Roy. Soc. Lon., 203, p. 385, 1904.

[45] G.S. Agarwal, and R. Inguva, "Effective medium theory of a heterogeneous medium with individual grains having a nonlocal dielectric function", Phys. Rev. B 30, p. 6108, 1984.

[46] Scott Kirkpatrick, "Percolation and Conduction", Rev. Mod. Phys., 45, p. 574, 1973.

[47] Scott Kirkpatrick, "The Geometry of the Percolation Threshold", AIP Conf. Proc. No. 40 on Electrical Transport and Optical Properties of Inhomogeneous Media,, J.C. Garland and D.B. Turner, eds., A.I.P. Pub., New York, p. 99, 1978.

[48] Vera V. Daniels, <u>Dielectric</u> <u>Relaxation</u>, Academic Press, London, 1967.

[49] Itzhak Webman, J. Jortner, and M. Cohen, "Theory of optical and microwave properties of microscopic inhomogeneous materials", Phys. Rev., B., 15, p. 5712, 1977.

[50] Stephen Wallin, John Kosinski, and Arthur Ballato, Numerical Analysis of Composite Dielectrics: Preliminary Results for Two Dimensions, Tech. Report SLCET-TR-90-11, Elec. Tech. and Dev. Lab., U.S. Army Lab. Command, Ft. Monmouth, NJ 07703, 1990.

[51] Th.G. Scholte, "A Contribution to the Theory of the Dielectric Constant of Polar Liquids", Physica, 15(5-6), pp. 437-450, July 1949.

[52] M. Mandel, "The Dielectric Constant and Maxwell-Wagner Dispersion of Oriented Prolate Spheroids", Physica, 27, pp. 827-840, 1961.

[53] B.P. Pradhan, and R.C. Gupta, "Dielectric Constants of Mixtures", Dielectrics, Feb. 1964.

[54] Leonard S. Taylor, "Dielectric Properties of Mixtures", IEEE Trans. Antennas and Prop., AP-13(6), pp. 943-947, Nov. 1965.

[55] C. Boned, and J. Peyrelasse, "Some comments on the complex permittivity of ellipsoids dispersed in continuum media", J. Phys. D: Appl. Phys. (UK), 16, pp. 1777-1784, 1983.

APPENDIX A. Depolarization

The resultant permittivity of a dielectric ellipsoid of one permittivity embedded in a medium of another permittivity can be determined by solving Laplace's equation within each of the media using confocal ellipsoidal harmonics [16-18, 30, 31, 51-55]. The solutions for the two regions are then matched at the dielectric interface according to the boundary conditions of continuity of the normal component of the electric displacement field and continuity of the tangential component of the electric field. A geometric factor related to the ellipsoid shape which emerges from the solution is

$$A_p = \frac{abc}{2} \int_o^\infty \frac{du}{R\ (u+p^2)} \qquad (p=a,b,c)$$

where a, b, and c are the ellipsoid axes and A_p is the depolarization factor when the overall fields are aligned along an axis. Accordingly, within the integral p assumes the value of a, b, or c. The function R is defined by positive $R^2 = (x+a^2)(x+b^2)(x+c^2)$. Integration by parts yields the identity $A_a+A_b+A_c = 1$. Thus the relation for the case a=b=c is A =1/3. The two-dimensional case is obtained by examining the limit as one of the ellipse axes goes to infinity so as to effectively eliminate it from the integral. Thus in the two-dimensional case the identity is $A_a+A_b = 1$ and for a=b we have A = 1/2. The same procedure can be applied to solve the one-dimensional case giving the trivial solution of A = 1. In general, a spheroid (ellipsoid with a=b=c) has a depolarization of A = 1/N, where N is the number of spatial dimensions. Other oblate and prolate cases can also be evaluated directly with the depolarization integral.

APPENDIX B. Program Codes

```
 - * - * - * - * - * - * - * - * - * - * - * - * - * -
                         "DIEL_BOND"
    Program in HP BASIC for numeric analysis based on the bond model
 - * - * - * - * - * - * - * - * - * - * - * - * - * -

10    !  <  <  <  <  <  <  <  "DIEL_BONDS"  >  >  >  >  >  >  >
20    ! * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - *
30    !  A main program to evaluate a 3 dimensional composite complex
40    !  dielectric response for a pixel network of capacitors.
50    !                                            S. R. Wallin, 6/1/91
60    ! * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - *
70    PRINT " MEMORY IS";VAL(SYSTEM$("AVAILABLE MEMORY"))/8;"(reals)"
80    OPTION BASE 1
90    DATA 1,0,2,1,4,3,7,6,11,10,16,15,22,21,29,28,37,36 ! Prog diel data
100   COM /Pass/Relay(0:7)                  ! for sharing to subs 1 var
110   COM /Pixel/Chdr$[80],Dhdr$[80],INTEGER Lxtnt,Pixl(1:20,1:20,1:20)
120   COMPLEX Hpiv(1:202),Hpr(1:10000)      ! dim to reasonable size for 3D
130   DIM Hdr$[80]                          ! available string for headers
140   COMPLEX Admt(0:7),Adms1,Adms2         ! neighbor admittance values
150   INTEGER Kube,Xt(0:7),Yt(0:7),Zt(0:7)  ! neighbor addresses
160   COM /Memr/Graf(1:512,1:4),Ahdr$[80],Bhdr$[80],INTEGER Rep,Kwd  !trials mem
170   !***> COM areas can be reaccessed with next RUN if identical name & sizes
180   !***> nb., max Lside >= .5 + cube root(.25 + 2*(max dim - 1) )
190   LET Start=TIMEDATE
200   INTEGER Lside,Kond,Nodesz
210   INTEGER Ptrn,Nd,Nd1,Nd2,Boxes,Slant,Sprss
220   INTEGER Xkin,Ykin,Zkin,Xcnt,Ycnt,Zcnt,Xaddr,Yaddr,Zaddr
230   INTEGER Qdrnt,Rptr,Trans,Pose,Grpt,Tls,Sctr,Itmp,Occp,Nsvf,Rsw
240   INTEGER Hmem,Hedge,Hpremax,Hopped,Hsteps,Hnde,Hcnt,Kcnt,Hkm
250   INTEGER Cnmb,Hnbr(0:7),Hcnr,Jxt
260   DIM Frpx(0:9),Msd$[60],Fln$[60],Fsv$[60]
270   COMPLEX Ctmp,Resp,Hnrm,Diel(1:9)
280   LET Grpt=0                         ! Initialize the data storage counter
290   ! * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - *
300   !   for which the integer variables roles are:
310   !    Relay = an available common pass variables
320   !    Lside = the # of pixel capacitor elements encounter along an edge
330   !            of the square of pixels
340   !    Tls = Lside or Lside/2 if 2x2x2 tiling
350   !    Qdrnt =  quadrant pixel array expanding switch, 0=off & 1=on
360   !    Px_tot = total # of pixels in square = Lside*Lside
370   !    Rsw = even/odd switch, etc
380   !    Kond = the boundary condition on the sides of the overall composite
390   !           capacitor, 1) insulating sides or 2) periodic or sides which
400   !           wrap around
410   !    Nodesz = the maximum number of interaction nodes in forming
420   !             network, with 0 as the ground or base plate, 1 as center
430   !             node, and the final node number for the top plate.
440   !             Its value is: L*L*L/4+L*L/4-5L/2+4
450   !
460   !                 X SECTION OF CAPACITOR CUBE
470   !                            |
```

```
480   !                             |
490   !                   ==================== top node or plate
500   !                    X X X X X X X X X X
510   !                    X X X X X X X X X X
520   !                    X X X X X X X X X X
530   !                    X X X X X X X X X X
540   !                    X X X X X X X X X X  center node at midpoint
550   !                    X X X X X X X X X X
560   !                    X X X X X X X X X X
570   !                    X X X X X X X X X X
580   !                    X X X X X X X X X X
590   !                   ==================== base node or plate
600   !                           __|__
610   !                            ___
620   !                             _              ...  X's represent nodes
630   !
640   !     Pixl() = the overlaying matrix representing the capacitor pixels
650   !     Dsplc() = displacement current of pixel per normalized volt/meter
660   !     Potnt() = pixel voltage relative to one volt across entire sample
670   !     Diel() = dielectric value or admittance value of a capacitor pixel
680   !              attached to addresses represented in the pixel grid
690   !     Frpx() = volume fractions associated with pixel types
700   !     Xt(),Yt() = neighbor addresses
710   !     Fln$ = string refering to a filename, Hdr$ = 80 chars
720   !     Ahdr$,Bhdr$ = headers of 80 chrs for Data Title & ID for COM /Memr/
730   !     Chdr$,Dhdr$ = headers of 80 chrs for Pixel Title & ID for COM /Pixel/
740   !     Ptrn = choice of pixel grid filling pattern
750   !     Nd = a single number label for a node
760   !     Nd1,Nd2 = refers to a 1st node & a 2nd node @ specified by single
770   !               node numbers
780   !     Xkin,Ykin,Zkin = kinship 3D address of a node number ie (x,y,z)
790   !     Xcnt,Ycnt,Zcnt = step counters to pixels neighbouring a node in 3D
800   !     Xaddr,Yaddr,Zaddr = addresses of neighbouring pixels in 3D
810   !     Boxes = total concentric boxes fitting within pixel grid or
820   !             number of 2x2 cell blocks along an edge on pixel grid
830   !     Jxt = a juxtaposition counter
840   !     Kube = inside cube territory test
850   !     Slant = 0 if foward slash or 1 if backslash slanting capacitor
860   !     Rep,Rptr = overall number of repeats, Kwd=# of data storage types
870   !     Grpt = overall plus transpose repeats for use of data storage
880   !     Sprss = Suppression of printout details
890   !     Trans,Pose = Pixel transpose selection
900   !     Resp = Overall dielectric response of pixel sample along E
910   !     Nsvf = switch indicating if intended to save repeat info
920   !     Tmp,Tmp1,Tmp2,Vt1,Vt2 = reals available for various uses
930   ! * - * - * - * - * - * - * - *   * - * - * - * - * - * - * - * - * - *
940   PRINT
950   PRINT " > > > Happy capacitor composite adventures in 3 dimensions < < <"
960   PRINT "                preformed on ";DATE$(TIMEDATE);
970   PRINT " at ";TIME$(TIMEDATE)
980   PRINT
990   ! #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
1000  ! The hopper reduction subarray:                     S. Wallin, July 1990
1010  !       ____  .  .  .  .  .  .  .  .  .  .  > large symmetric sparse matrix .
1020  !     | 1,1|     <=< NODE PAIR                                              .
```

```
1030  !      |_#1_|_____  <=<hopper address                                   .
1040  !      | 2,1| 2,2|                                                       .
1050  !      |_#2_|_#3_|____                                                   .
1060  !      | 3,1| 3,2| 3,3|                                                  .
1070  !      |_#4_|_#5_|_#6_|                                                  .
1080  !      .                 ..        The hopper subarray moves             .
1090  !      .                     ..    down as pivoting progresses           .
1100  !    \/ lg sym sparse matrix        ..                                   .
1110  !      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
1120  ! The "H" prefix is mainly used to denote variable use in hopper program
1130  ! Hnbr(*) = neighbor nodes
1140  ! Hedge = altitude or base or diag # elements of reduction hopper
1150  ! Hmem = the total # of elements contained within reduction hopper
1160  ! Hpremax = same as Hmem but less Hedge (ie less largest row)
1170  ! Hsteps = extent of larger interaction matrix upon hopper reduces
1180  ! Hcnr = corner node #s of insulated sides Pixel grid cube
1190  ! Hpiv() = Pivoting vector of node reduction
1200  ! Hpr() = working hopper array of matrix reduction
1210  !  #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
1220  PRINT ">Try hopping along to a solution of sparse matrices at ";
1230  DISP "IO to be: 0)default 1)lab 3.5"" 2)lab hardisc 3,4)A,B office ";
1240  INPUT "5)user defined",Nd
1250  IF Nd<0 THEN STOP
1260  IF Nd=0 THEN Msd$=""
1270  IF Nd=1 THEN Msd$=":CS80,700,1"
1280  IF Nd=2 THEN Msd$=":CS80,700"
1290  IF Nd=3 THEN Msd$=":CS80,703,0"
1300  IF Nd=4 THEN Msd$=":CS80,703,1"
1310  IF Nd=5 THEN INPUT "Name (completely) storage?",Msd$
1320  IF Msd$<>"" THEN PRINT RPT$(" ",50);"storage";Msd$
1330  PRINT " The pattern choices are:"
1340  PRINT "  0) internal, via COM /Memr/"
1350  PRINT "  1) from file storage"
1360  PRINT "  2) every pixel filled by user"
1370  PRINT "  3) random (i.e. well mixed)"
1380  PRINT "  4) by slanted fill level"
1390  PRINT "  5) with  an circle or ellipse of which can be tilted"
1400  PRINT "  6) with strata"
1410  PRINT "  7) concentric boxes"
1420  PRINT "  8) an ellipse with host & inclusion (2 components only,";
1430  PRINT " but symm wrt 1/2 vol)"
1440  INPUT "Select design of pixel grid? (see above)",Ptrn
1450  IF Ptrn<0 THEN STOP
1460 !DISP " & boundary conditions? 1) Insulative ";
1470 !INPUT "2) Wrap around or periodic",Kond
1480  LET Kond=1 !TEMPORARY
1490  IF Kond<0 THEN STOP
1500  IF Kond=0 THEN Kond=1
1510  IF Kond>2 THEN Kond=1+BIT(Kond+1,0)
1520 !INPUT "Use 2x2x2 tiles on pixel grid? 0) No 1) Yes",Tls
1530  Tls=0 !temporary until programmed
1540  IF Tls<0 THEN STOP
1550  LET Tls=1+(Tls=1)
1560 !INPUT "Oct- fold symmetry expansion of pixel grid? 0) No 1) Yes",Qdrnt
1570  LET Qdrnt=0 !temporary until programmed
```

```
1580  IF Qdrnt<0 THEN STOP
1590  LET Qdrnt=1+(Qdrnt=1)
1600  SELECT Ptrn
1610  CASE =0
1620    IF Lxtnt<2 THEN
1630      PRINT " Are Pixels there in memory? ..idled ..start again"
1640      STOP
1650    END IF
1660    IF Lxtnt>21 THEN PRINT " .. there may be too many Pixels"
1670    REDIM Pixl(1:Lxtnt,1:Lxtnt,1:Lxtnt)
1680    LET Lside=Lxtnt*Qdrnt*Tls
1690    PRINT " From internal memory via COM, Pixels";Lxtnt;"x";Lxtnt;"x";Lxtnt;",
                        title,"
1700    IF Chdr$<>"" THEN PRINT Chdr$
1710    IF Dhdr$<>"" THEN PRINT Dhdr$
1720  CASE =1                                  ! Get pixels from file
1730    INPUT " Enclose (in """"s) file name to contain pixel pattern?",Fln$
1740    IF Fln$="" THEN STOP
1750    IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
1760    DISP " File named """;Fln$;""" ([";LEN(Fln$);"] characters)";
1770    DISP " is being read from storage"
1780    ASSIGN @Pixsrc TO Fln$;FORMAT OFF
1790    ENTER @Pixsrc;Chdr$,Dhdr$;Lxtnt      ! NB header assigned length of 80
1800    PRINT " Pixels contained in file """;Fln$;""", entitled with"
1810    PRINT Chdr$
1820    PRINT Dhdr$
1830    REDIM Pixl(1:Lxtnt,1:Lxtnt,1:Lxtnt) ! read initial Pixl(*) array
1840    ENTER @Pixsrc;Pixl(*)               ! retreive pixels from file
1850    ASSIGN @Pixsrc TO *                 ! close file
1860    LET Lside=Lxtnt*Qdrnt*Tls           ! actual Pixel side anticipated
1870    PRINT
1880  CASE ELSE                             ! Generate pixels
1890    DISP "How big a capacitor pixel grid in elements/side? ";
1900    INPUT "(even #, max ~20 int addr lmt)",Lside
1910    IF Lside<0 THEN STOP
1920    IF Lside<>SHIFT(SHIFT(Lside,1),-1) THEN
1930      PRINT " Odd";Lside;"Pixel length changed to even";
1940      LET Lside=SHIFT(SHIFT(Lside,1),-1)
1950      PRINT Lside
1960    END IF
1970    IF Lside=0 THEN Lside=2
1980    LET Lxtnt=Lside                      ! initial Pixel side length
1990    LET Lside=Lside*Qdrnt*Tls            ! Pixel side length anticipated
2000    IF Lside>32 THEN PRINT " ... near integer addressing limit"
2010  END SELECT                             ! end Ptrn test
2020  PRINT
2030  IF Tls=2 OR Qdrnt=2 THEN PRINT " Pixels now
                        measures";Lside;"x";Lside;"x";Lside
2040  PRINT " Pattern=";Ptrn;")";Lside;"x";Lside;"x";Lside
2050  IF Tls=1 THEN PRINT "pixels,";
2060  IF Tls=2 THEN PRINT "tiled pixels,";
2070  IF Kond=1 THEN PRINT " insulated or ""D"" field parallel to edge."
2080  IF Kond=2 THEN PRINT " periodic or voltage wrapping around at edges."
2090 !ALLOCATE REAL Dsplc(Lside,Lside,Lside),Potnt(Lside,Lside,Lside)
2100  !***> Initializing
```

```
2110  MAT Diel=(CMPLX(0,0))
2120  DISP "Dielectric sources? ";
2130  DISP "1) (from program) 2) user ";
2140  INPUT "3) by steps 4) lossy/real binary ",Nd
2150  IF Nd=0 THEN Nd=2
2160  SELECT Nd
2170  CASE =1
2180    FOR Nd1=1 TO 9
2190      READ Diel(Nd1)
2200    NEXT Nd1
2210  CASE =2
2220    FOR Nd1=1 TO 9
2230      DISP "Give dielectric complex value at";Nd1;
2240      INPUT "? (or enter (-real,0) if to cease)",Diel(Nd1)
2250      IF REAL(Diel(Nd1))<0 THEN
2260        LET Diel(Nd1)=CMPLX(0,0)
2270        LET Nd1=9
2280      ELSE
2290        PRINT "diel[";Nd1;"]=(";VAL$(DROUND(REAL(Diel(Nd1)),4));",";
2300        PRINT VAL$(DROUND(IMAG(Diel(Nd1)),4));"),";
2310      END IF
2320    NEXT Nd1
2330  CASE =3
2340    INPUT "Complex dielectric value of pixel type ""[1]""?",Ctmp
2350    DISP "Multiplier of progression for each succeeding value ";
2360    INPUT "to fill [2],[3],..,[9] ?",Tmp1
2370    FOR Nd1=1 TO 9
2380      Diel(Nd1)=Ctmp
2390      Ctmp=Ctmp*Tmp1
2400      PRINT "diel[";Nd1;"]=(";VAL$(DROUND(REAL(Diel(Nd1)),4));",";
2410      PRINT VAL$(DROUND(IMAG(Diel(Nd1)),4));"),";
2420    NEXT Nd1
2430  CASE =4
2440    DISP "Constituent [1] is solely imaginary(lossy), ie=(0,_),";
2450    INPUT " what is the value? ",Tmp1
2460    LET Diel(1)=CMPLX(0,Tmp1)
2470    DISP "Constituent [2] is solely real, ie=(_,0),";
2480    INPUT " what is the value? ",Tmp1
2490    LET Diel(2)=CMPLX(Tmp1,0)
2500  CASE ELSE
2510    STOP
2520  END SELECT
2530  PRINT
2540  LET Sprss=1
2550  IF Nodesz<32 THEN
2560    INPUT "Surpress screen listing details, 0) No 1) Yes?",Sprss
2570    IF Sprss<0 THEN STOP
2580  END IF
2590  INPUT "Any overall repeats (0=single manual run)? ",Rep
2600  LET Rep=SHIFT(Rep,-1)                !! double if conjugates
2610  LET Relay(0)=Rep                     ! Manual when Relay=0
2620  LET Relay(1)=SHIFT(Rep,1)+1          ! repeat conjugates
2630  IF Rep<0 THEN STOP
2640  IF Rep=0 THEN LET Rep=1               ! If manual, still go thru once
2650  LET Nsvf=0
```

```
2660 IF Rep>1 THEN
2670   INPUT "Save data in a file? 0=No 1=Yes ",Nsvf
2680   LET Nsvf=BIT(Nsvf,0)
2690   IF Nsvf=0 THEN
2700     LET Fsv$=""
2710   ELSE
2720     INPUT "Name a new file to receive data output ",Fsv$
2730     IF Fsv$<>"" THEN LINPUT "Title or description? (<80) ",Ahdr$
2740   END IF
2750 END IF
2760 INPUT "Desire transpose of pixel grid? 0) No 1) Yes",Pose
2770 IF Pose<0 THEN STOP
2780 LET Pose=1+BIT(Pose,0)
2790 REM " Solution acheived by a sparse matrix reduced pivoting technique"
2800 !***>> Overall repetition, may require additional editing
2810 LET Stmrp=TIMEDATE                     ! Timer for repeat"
2820 LET Kwd=4             ! user has selected to program for 4 data columns
2830 REDIM Graf(1:Rep,1:Kwd)
2840 MAT Graf=(0)                           ! Intialize to zero
2850 FOR Rptr=1 TO Rep
2860   LET Rsw=BIT(Rptr,0)                  !Even=0/Odd=1 switch
2870 !!LET Grpt=Grpt+1                      ! Increment data store counter
2880   LET Grpt=Grpt+Rsw                   !!Increment storage counter(on 2s)
2890   IF Relay(0)<>0 THEN LET Relay(0)=Rptr !Standard option of Relay
2900   MAT Frpx=(0)                         !Reset volume fractions to 0
2910   MAT Diel=CONJG(Diel)                 !!conjugation option
2920 !!IF Rptr>0 AND Rptr MOD Lside*Lside=0 THEN MAT Diel=CONJG(Diel)!!Conjg
2930   !**> if Ptrn=0 internal or Ptrn=1 then Pixels  read from file
2940   IF Ptrn=2 THEN CALL Pixl3d_fill
2950   IF Ptrn=3 THEN CALL Pixl3d_rand
2960   IF Ptrn=4 THEN CALL Pixl3d_tilt
2970   IF Ptrn=5 THEN CALL Pixl3d_ellps
2980   IF Ptrn=6 THEN CALL Pixl3d_strat
2990   IF Ptrn=7 THEN CALL Pixl3d_cbox
3000   IF Ptrn=8 THEN CALL Pixl2d_3d
3010   IF Tls=2 THEN
3020     LET Xkin=SIZE(Pixl,1)              ! redimensioning
3030     LET Ykin=SIZE(Pixl,2)
3040     LET Zkin=SIZE(Pixl,3)
3050     LET Nd1=2*(Xkin+Ykin+Zkin) DIV 3 ! ave new dimension
3060     LET Nd2=SHIFT(Xkin*Ykin*Zkin,-3) ! 8*Xkin*Ykin*Zkin
3070     REDIM Pixl(1:1,1:Nd2)              ! shifting Pixl array contents
3080     FOR Xcnt=(Xkin-1) TO 0 STEP -1
3090       FOR Ycnt=Ykin TO 1 STEP -1
3100         LET Pixl(1,1,Xcnt*2*Ykin+Ycnt)=Pixl(1,1,Xcnt*Ykin+Ycnt)
3110       NEXT Ycnt
3120     NEXT Xcnt
3130     REDIM Pixl(1:Nd1,1:Nd1,1:Nd1)      ! set new array dimen
3140     FOR Xcnt=Xkin TO 1 STEP -1         ! tiling 2x2
3150       FOR Ycnt=Ykin TO 1 STEP -1
3160         !LET Itmp=Pixl(1,Xcnt,Ycnt)
3170         LET Xaddr=SHIFT(Xcnt,-1)       ! effective 2* op
3180         LET Yaddr=SHIFT(Ycnt,-1)
3190         LET Pixl(1,Xaddr,Yaddr)=Itmp
3200         LET Pixl(1,Xaddr-1,Yaddr)=Itmp
```

```
3210          LET Pixl(1,Xaddr,Yaddr-1)=Itmp
3220          LET Pixl(1,Xaddr-1,Yaddr-1)=Itmp
3230        NEXT Ycnt
3240      NEXT Xcnt
3250    END IF
3260    IF Qdrnt=2 THEN
3270      LET Nd1=(SIZE(Pixl,1)+SIZE(Pixl,2)+SIZE(Pixl,3)) DIV 3
3280      LET Nd2=SHIFT(Nd2,-1)
3290      REDIM Pixl(1:Nd2,1:Nd2,1:Nd2)     ! redim to dble quad duplic
3300      FOR Xcnt=Nd1 TO 1 STEP -1
3310        LET Xaddr=Nd2+1-Xcnt            ! quad complement X counter
3320        LET Xkin=SHIFT(Xcnt+1,1)        ! effectively DIV 2 op
3330        LET Ykin=BIT(Xcnt+1,0)          ! effectively odd<=>even op
3340        FOR Ycnt=1 TO Nd1
3350          LET Yaddr=Nd2+1-Ycnt          ! quad complement Y counter
3360          LET Itmp=Pixl(1,Xkin,Ycnt+Nd1*Ykin)
3370          LET Pixl(1,Xcnt,Ycnt)=Itmp   ! Itmp takes care of redim elements
3380          LET Pixl(1,Xaddr,Ycnt)=Itmp
3390          LET Pixl(1,Xcnt,Yaddr)=Itmp
3400          LET Pixl(1,Xaddr,Yaddr)=Itmp
3410        NEXT Ycnt
3420      NEXT Xcnt
3430    END IF
3440    LET Lside=SIZE(Pixl,1)            ! update Pixl extent along edge
3450    LET Px_tot=Lside*Lside*Lside      ! update actual Pixl volume
3460    LET Boxes=SHIFT(Lside,1)          ! 1/2 of edge length in Pixls
3470    IF Boxes<1 THEN PRINT "WARNING! may be too small of a pixel grid"
3480    LET Nodesz=SHIFT(Lside*Lside*(Lside+1),2)-SHIFT(5*Lside,1)+4
3490    !***> Tranpose of Pixel grid
3500    LET Trans=Pose ! if loop then use next line
3510   !FOR Trans=1 TO Pose
3520    IF Trans=2 THEN                   ! Tranpose in @ Xplanes
3530      PRINT " TRANSPOSING about the X direction"
3540      FOR Xcnt=1 TO Lside
3550        FOR Ycnt=1 TO Lside
3560          FOR Zcnt=(Ycnt+1) TO Lside
3570            LET Itmp=Pixl(Xcnt,Zcnt,Ycnt) !swap Ycoordinate<->Zcoordinate
3580            LET Pixl(Xcnt,Zcnt,Ycnt)=Pixl(Xcnt,Ycnt,Zcnt)
3590            LET Pixl(Xcnt,Ycnt,Zcnt)=Itmp
3600          NEXT Zcnt
3610        NEXT Ycnt
3620      NEXT Xcnt
3630    END IF
3640    !***> Evaluation of pixel type volume fractions
3650    IF NOT (Sprss) OR Lside<10 THEN PRINT "  Pixels";Lside;"x";Lside;"x";Lside
3660    FOR Xcnt=1 TO Lside              !by X-plane sections
3670      IF NOT (Sprss) OR Lside<10 THEN PRINT RPT$(" ",Boxes);"X=";VAL$(Xcnt)
3680      FOR Zcnt=Lside TO 1 STEP -1    !Z printout by rows, largest Z 1st row
3690        FOR Ycnt=1 TO Lside          !Y printout across, increasing Y
3700          LET Frpx(Pixl(Xcnt,Ycnt,Zcnt))=Frpx(Pixl(Xcnt,Ycnt,Zcnt))+1
3710          IF NOT (Sprss) OR Lside<10 THEN PRINT " ";VAL$(Pixl(Xcnt,Ycnt,Zcnt));
3720        NEXT Ycnt
3730        IF NOT (Sprss) OR Lside<10 THEN PRINT
3740      NEXT Zcnt
3750    NEXT Xcnt
```

```
3760    MAT Frpx=Frpx/(Px_tot)
3770    PRINT " Volume %s: ";
3780    FOR Nd=1 TO 9
3790      IF Frpx(Nd)<>0 THEN PRINT PROUND(100*Frpx(Nd),-1);"%=>[";Nd;"],";
3800    NEXT Nd
3810    PRINT
3820    IF Frpx(0)<>0 THEN PRINT "WARNING! check pixels"
3830    DISP " .. wait";Rptr;"of";Rep;".. solving node interact matrix,";
3840    DISP Nodesz;"by";Nodesz;"from time ";TIME$(TIMEDATE)
3850    LET Tmp=TIMEDATE                     ! Benchmarker
3860   !***>  HOP technique of sparse matrix reduction
3870    PRINT " Solving node INTERACTION matrix";Nodesz;"x";Nodesz;"via hopper"
3880    LET Hedge=SHIFT(Lside*Lside,1)+Lside-2 !hopper edge, max interact diff
3890    LET Hmem=(1.0+Hedge)*Hedge/2         ! total hopper memory requirement
3900    LET Hsteps=Nodesz
3910    LET Hpremax=Hmem-Hedge
3920    REDIM Hpiv(1:Hedge),Hpr(1:Hmem)
3930    MAT Hpiv=(CMPLX(0,0))
3940    MAT Hpr=(CMPLX(0,0))
3950    LET Hkm=0                            ! 0,1,3,6.. previous hopper row end
3960    FOR Hnde=1 TO Hedge                  ! Filling hopper work array
3970      IF NOT (Sprss) THEN PRINT " node's";Hnde;" lower neighbors are";
3980      FOR Sctr=0 TO 7                    ! Diagonal or self interact terms
3990        IF Kond=1 THEN CALL Cv3ndi(Hnde,Lside,Sctr,Xt(Sctr),Yt(Sctr),Zt(Sctr))
4000        IF Kond=2 THEN CALL Cv3ndp(Hnde,Lside,Sctr,Xt(Sctr),Yt(Sctr),Zt(Sctr))
4010        LET Kube=(Xt(Sctr)<1 OR Xt(Sctr)>Lside OR Yt(Sctr)<1) !in cube?
4020        LET Kube= NOT (Kube OR Yt(Sctr)>Lside OR Zt(Sctr)<1 OR Zt(Sctr)>Lside)
4030        IF Kube THEN
4040          LET Admt(Sctr)=Diel(Pixl(Xt(Sctr),Yt(Sctr),Zt(Sctr)))
4050          IF Kond=1 THEN              ! adj for insl BC on Pixel grid
4060            LET Hcnr=(Xt(Sctr)=1 OR Xt(Sctr)=Lside)!Pix on corner
4070            LET Hcnr=(Hcnr AND (Yt(Sctr)=1 OR Yt(Sctr)=Lside))
4080            LET Hcnr=(Hcnr AND Zt(Sctr)>1 AND Zt(Sctr)<Lside)
4090            IF Hcnr THEN              ! test if corner
4100              LET Jxt=SHIFT(SHIFT(Zt(Sctr),1),-1)+BIT(BINCMP(Zt(Sctr)),0)
4110              LET Adms1=Diel(Pixl(Xt(Sctr),Yt(Sctr),Zt(Sctr)))
4120              LET Adms2=Diel(Pixl(Xt(Sctr),Yt(Sctr),Jxt))
4130              LET Adms1=Adms1*Adms2/(Adms1+Adms2)
4140              LET Admt(Sctr)=Adms1  ! admittance of path
4150            END IF                    ! end of Hnde=Hcnr test
4160          END IF                      ! end if for Kond=1  test
4170          LET Hpr(Hkm+Hnde)=Hpr(Hkm+Hnde)+[illegible](Sctr) !self interact
4180          IF BIT(Sctr,0)=0 THEN       ! find out lower Z neighbors
4190            IF NOT (Sprss) THEN
4200              PRINT " (";VAL$(Xt(Sctr));",";VAL$(Yt(Sctr));",";
4210              PRINT VAL$(Zt(Sctr));")@";
4220            END IF
4230            SELECT Kond
4240            CASE =1
4250              LET Hnbr(Sctr)=FNNni(Xt(Sctr),Yt(Sctr),Zt(Sctr),Lside,1)
4260            CASE =2
4270              LET Hnbr(Sctr)=FNNnp(Xt(Sctr),Yt(Sctr),Zt(Sctr),Lside,1)
4280            CASE ELSE
4290              PRINT " out of bounds, boundary condition, detected in HOPper"
4300            END SELECT
```

```
4310            IF NOT (Sprss) THEN PRINT VAL$(Hnbr(Sctr));
4320            IF Hnbr(Sctr)>Hnde THEN PRINT " ???";
4330            IF Hnbr(Sctr)<=Hnde-Hedge THEN PRINT "Node";Hnbr(Sctr);"outside of
                    hopper at";Hnde
4340            IF Hnbr(Sctr)<Hnde AND Hnbr(Sctr)>0 THEN
4350              LET Hpr(Hkm+Hnbr(Sctr))=Hpr(Hkm+Hnbr(Sctr))-Admt(Sctr)!neighbr
4360            END IF
4370          END IF                    ! end if for even Sctr
4380        END IF                      ! end if for Kube
4390      NEXT Sctr
4400      IF NOT (Sprss) THEN PRINT     ! end of output line
4410      LET Hkm=Hkm+Hnde              ! loop count accumulator
4420    NEXT Hnde                       ! end of set up of work matrix
4430    LET Kcnt=0
4440    IF NOT (Sprss) THEN
4450      FOR Xcnt=1 TO Hedge           ! printout HOPper
4460        FOR Ycnt=1 TO Xcnt
4470          PRINT "(";VAL$(DROUND(REAL(Hpr(Kcnt+Ycnt)),3));",";
4480          PRINT VAL$(DROUND(IMAG(Hpr(Kcnt+Ycnt)),3));") ";
4490        NEXT Ycnt
4500        PRINT
4510        LET Kcnt=Kcnt+Xcnt
4520      NEXT Xcnt
4530    END IF
4540    FOR Hopped=1 TO Hsteps-1        ! Let the PIVOTING begin, & drip dry
4550      LET Hnde=Hedge+Hopped         ! count of oncoming node number
4560      LET Hpiv(1)=CMPLX(1,0)        ! normalize to 1st elment pivot vectr
4570      LET Kcnt=2                    ! convert array storage for 1st colmn
4580      IF Hpr(1)=CMPLX(0,0) THEN     ! don't waste steps if 0
4590        PRINT "1st=0?";
4600        MAT Hpiv=(CMPLX(0,0))
4610      ELSE
4620        LET Hnrm=1/Hpr(1)           ! set normalizing multiplier
4630        FOR Hcnt=2 TO Hedge         ! set pivot vector
4640          LET Hpiv(Hcnt)=Hpr(Kcnt)*Hnrm
4650          LET Kcnt=Kcnt+Hcnt        ! 2,4,7,11,..@1st elmnt in hoppr row
4660        NEXT Hcnt
4670      END IF
4680      !***> one can output pivot here for backsub later
4690      IF NOT (Sprss) THEN
4700        IF Hnde<=Hsteps+1 THEN
4710          PRINT "feed node";Hnde-1;
4720          FOR Ycnt=1 TO Hedge     ! printout of hopper feed row
4730            PRINT "(";VAL$(DROUND(REAL(Hpr(Hpremax+Ycnt)),3));",";
4740            PRINT VAL$(DROUND(IMAG(Hpr(Hpremax+Ycnt)),3));") ";
4750          NEXT Ycnt
4760          PRINT
4770        END IF                      ! end if for Hnde<Hsteps
4780        PRINT " At reduction";Hopped;"the complex pivots are:"
4790        LET Xcnt=MIN(Hedge,Hsteps-Hopped+1) !significant
4800        FOR Hcnt=Xcnt TO 1 STEP -1
4810          PRINT "{";VAL$(DROUND(REAL(Hpiv(Hcnt)),4));",";
4820          PRINT VAL$(DROUND(IMAG(Hpiv(Hcnt)),4));"}";
4830        NEXT Hcnt
4840        IF Hpiv(1)=CMPLX(0,0) THEN PRINT "?!";
```

```
4850        PRINT
4860      END IF
4870      LET Kcnt=1                    ! initialize filling counter
4880      FOR Xcnt=2 TO Hedge           ! heart of pivoting
4890        IF Hpiv(Xcnt)<>CMPLX(0,0) THEN ! sparseness efficiency =0, no-op
4900          FOR Ycnt=2 TO Xcnt        ! adj each array row with pivot vectr
4910            IF Hpiv(Ycnt)<>CMPLX(0,0) THEN ! sparseness efficiency =0, no-op
4920              LET Hpr(Kcnt+Ycnt)=Hpr(Kcnt+Ycnt)-Hpr(Kcnt+1)*Hpiv(Ycnt)
4930            END IF
4940          NEXT Ycnt
4950        END IF
4960        LET Kcnt=Kcnt+Xcnt          !1,3,6,10,.. gives previous row end
4970      NEXT Xcnt
4980      LET Kcnt=0                    ! initialize filling counter lower
4990      FOR Xcnt=1 TO Hedge-1         ! X,Y refer to hopping counters
5000        FOR Ycnt=1 TO Xcnt          ! hopping along for shake up
5010          LET Hpr(Kcnt+Ycnt)=Hpr(Kcnt+Ycnt+1+Xcnt)
5020        NEXT Ycnt
5030        LET Kcnt=Kcnt+Xcnt
5040      NEXT Xcnt
5050      FOR Ycnt=1 TO Hedge           ! feed hopper, clear last row
5060        LET Hpr(Hpremax+Ycnt)=CMPLX(0,0)
5070      NEXT Ycnt
5080      IF NOT (Sprss) AND Hnde<Hsteps THEN PRINT " node's";Hnde;" lower neighbors
                  are";
5090      FOR Sctr=0 TO 7               ! find neighbors
5100        SELECT Hnde
5110        CASE <Hsteps                ! feed unless over lg array extent
5120          IF Kond=1 THEN CALL Cv3ndi(Hnde,Lside,Sctr,Xt(Sctr),Yt(Sctr),Zt(Sctr))
5130          IF Kond=2 THEN CALL Cv3ndp(Hnde,Lside,Sctr,Xt(Sctr),Yt(Sctr),Zt(Sctr))
5140          LET Kube=(Xt(Sctr)<1 OR Xt(Sctr)>Lside OR Yt(Sctr)<1) !in cube?
5150          LET Kube=(Kube OR Yt(Sctr)>Lside OR Zt(Sctr)<1 OR Zt(Sctr)>Lside)
5160          LET Kube= NOT (Kube)
5170          IF Kube THEN
5180            LET Admt(Sctr)=Diel(Pixl(Xt(Sctr),Yt(Sctr),Zt(Sctr)))
5190            IF Kond=1 THEN          ! adj for insl BC on Pixel grid
5200              LET Hcnr=(Xt(Sctr)=1 OR Xt(Sctr)=Lside)!Pix on corner
5210              LET Hcnr=(Hcnr AND (Yt(Sctr)=1 OR Yt(Sctr)=Lside))
5220              LET Hcnr=(Hcnr AND Zt(Sctr)>1 AND Zt(Sctr)<Lside)
5230              IF Hcnr THEN          ! test if corner
5240                LET Jxt=SHIFT(SHIFT(Zt(Sctr),1),-1)+BIT(BINCMP(Zt(Sctr)),0)
5250                LET Adms1=Diel(Pixl(Xt(Sctr),Yt(Sctr),Zt(Sctr)))
5260                LET Adms2=Diel(Pixl(Xt(Sctr),Yt(Sctr),Jxt))
5270                LET Adms1=Adms1*Adms2/(Adms1+Adms2)
5280                LET Admt(Sctr)=Adms1! admittance of path
5290              END IF                ! end of Hnde=Hcnr test
5300            END IF                  ! end if for Kond=1  test
5310            LET Hpr(Hmem)=Hpr(Hmem)+Admt(Sctr)
5320            IF BIT(Sctr,0)=0 THEN   ! find lower Z neighor interactions
5330              IF NOT (Sprss) THEN PRINT "
                     (";VAL$(Xt(Sctr));",";VAL$(Yt(Sctr));",";VAL$(Zt(Sctr));")@";
5340              SELECT Kond
5350              CASE =1
5360                LET Hnbr(Sctr)=FNNni(Xt(Sctr),Yt(Sctr),Zt(Sctr),Lside,1)
5370              CASE =2
```

```
5380                LET Hnbr(Sctr)=FNNnp(Xt(Sctr),Yt(Sctr),Zt(Sctr),Lside,1)
5390              END SELECT            ! to SELECT Kond
5400              IF NOT (Sprss) THEN PRINT VAL$(Hnbr(Sctr));
5410              IF Hnbr(Sctr)>=Hnde THEN PRINT " ???";
5420              IF Hnbr(Sctr)<=Hnde-Hedge THEN PRINT "Node";Hnbr(Sctr);"outside of
                        hopper at";Hnde
5430              IF Hnbr(Sctr)<Hnde AND Hnbr(Sctr)>0 THEN
5440                LET Hnbr(Sctr)=Hnbr(Sctr)+Hmem-Hnde
5450                LET Hpr(Hnbr(Sctr))=Hpr(Hnbr(Sctr))-Admt(Sctr)
5460              END IF
5470            END IF                    ! end if for even Sctr
5480          END IF                      ! end if in Kube
5490        CASE =Hsteps                  ! special case, exciter electrode
5500          IF BIT(Sctr,0)=0 THEN       ! do on even Sctr
5510           !IF NOT (Sprss) THEN PRINT " ";VAL$(Hsteps);
5520            FOR Xcnt=1 TO Boxes       ! pixels which abut exciter electrode
5530              LET Xkin=SHIFT(Xcnt,-1)+BIT(Sctr,2)-1! X address in Sctr
5540              FOR Ycnt=1 TO Boxes   ! & neighbors
5550                LET Ykin=SHIFT(Ycnt,-1)+BIT(Sctr,1)-1! Y addrss in Sctr
5560                LET Hpr(Hmem)=Hpr(Hmem)+Diel(Pixl(Xkin,Ykin,Lside))
5570                LET Hnbr(Sctr)=Hsteps-1-(Boxes-Ycnt+1)*Boxes+Xcnt
5580               !IF NOT (Sprss) THEN PRINT "@";VAL$(Hnbr(Sctr));!neighbr
5590                LET Hnbr(Sctr)=Hnbr(Sctr)+Hmem-Hnde !rel. to hopper address
5600                Hpr(Hnbr(Sctr))=Hpr(Hnbr(Sctr))-Diel(Pixl(Xkin,Ykin,Lside))
5610              NEXT Ycnt
5620            NEXT Xcnt
5630          END IF                      ! end if even SCTR
5640        END SELECT                    ! to SELECT Hnde
5650      NEXT Sctr
5660      IF NOT (Sprss) AND Hnde<=Hsteps THEN PRINT ! end of print out line
5670    NEXT Hopped
5680    IF NOT (Sprss) THEN
5690      PRINT " Complex hopper funnels down to {";DROUND(REAL(Hpr(1)),4);
5700      PRINT ",";DROUND(IMAG(Hpr(1)),4);"}"
5710    END IF
5720    !***> Hpr(1) contains the end of the interaction reduction
5730    !   LET Hpiv(Hsteps)=1/Hpr(1)      ! backsubstitute for solution vector
5740    !   FOR Hcnt=(Hsteps-1) TO 1 STEP -1
5750    !     LET Hpiv(Hcnt)=0
5760    !     FOR Kcnt=Hcnt TO Hsteps
5770    !      LET Hpiv(Hcnt)=Hpiv(Hcnt)-Pivotstorg(Hcnt,1+Kcnt-Hcnt)*Hpiv(Kcnt)
5780    !     NEXT Kcnt
5790    !   NEXT Hcnt
5800    PRINT "                ... at ";TIME$(TIMEDATE);" inversion excution ";
5810    PRINT "time took";PROUND(TIMEDATE-Tmp,-1);"seconds"
5820    DISP
5830    LET Resp=Hpr(1)/Lside                 ! principal diel resp
5840  !***> Pixl displacement field/current & potentials
5850   !  FOR Ycnt=1 TO Lside
5860   !    FOR Xcnt=1 TO Lside
5870   !      Slant=(Xcnt+Ycnt) MOD 2
5880   !      Xaddr=Xcnt-Boxes-Slant          ! (x,y) of node upper to pixel
5890   !      Yaddr=Ycnt-Boxes
5900   !      CALL Xy_to_node(Ndl,Xaddr,Yaddr,Lside,Kond)
5910   !      Xaddr=Xcnt-Boxes+Slant-1        ! (x,y) of node lower to pixel
```

```
5920  !       Yaddr=Ycnt-Boxes-1
5930  !       CALL Xy_to_node(Nd2,Xaddr,Yaddr,Lside,Kond)
5940  !       IF Nd1<>Nd2 AND Nd1>0 AND Nd2>0 THEN
5950  !         Dsplc(Xcnt,Ycnt)=Diel(Pixl(1,Xcnt,Ycnt))*(Hpiv(Nd1)-Hpiv(Nd2))
5960  !         Potnt(Xcnt,Ycnt)=(Hpiv(Nd1)+Hpiv(Nd2))/2
5970  !       END IF
5980  !       IF Nd2=0 AND Nd1>0 THEN
5990  !         Dsplc(Xcnt,Ycnt)=Diel(Pixl(1,Xcnt,Ycnt))*Hpiv(Nd1)
6000  !         Potnt(Xcnt,Ycnt)=Hpiv(Nd1)/2
6010  !       END IF
6020  !     NEXT Xcnt
6030  !   NEXT Ycnt
6040  !***> additional modification of Potential & Displacement array fields
6050  !   IF Kond=1 THEN
6060  !     FOR Ycnt=2 TO (Lside-2) STEP 2
6070  !       FOR Nd=-1 TO 1 STEP 2
6080  !         LET Xcnt=(Nd+1)*Boxes+(Nd=-1)  ! (Xcnt,Ycnt) refer to pixel
6090  !         LET Xaddr=Nd*(Boxes-1)         ! (Xaddr,Yaddr) refer to node
6100  !         LET Yaddr=Ycnt-Boxes
6110  !         CALL Xy_to_node(Nd1,Xaddr,Yaddr+1,Lside,Kond)! node # upper
6120  !         CALL Xy_to_node(Nd2,Xaddr,Yaddr-1,Lside,Kond)! node # lower
6130  !         IF Nd1<>Nd2 AND Nd1>0 AND Nd2>0 THEN ! Evaluate along side nodes
6140  !           LET Tmp1=Diel(Pixl(1,Xcnt,Ycnt+1))! Upper dielectric pixel
6150  !           LET Tmp2=Diel(Pixl(1,Xcnt,Ycnt))! Lower dielectric pixel
6160  !           LET Vt1=Hpiv(Nd1)
6170  !           LET Vt2=Hpiv(Nd2)
6180  !           IF Tmp1<>0 AND Tmp2<>0 THEN Tmp=(Vt1*Tmp1+Vt2*Tmp2)/(Tmp1+Tmp2)
6190  !           LET Potnt(Xcnt,Ycnt+1)=(Vt1+Tmp)/2! Pixl volts
6200  !           LET Potnt(Xcnt,Ycnt)=(Vt2+Tmp)/2
6210  !           IF Tmp1<>0 AND Tmp2<>0 THEN
6220  !             LET Dsplc(Xcnt,Ycnt+1)=(Vt1-Vt2)/(1/Tmp1+1/Tmp2)
6230  !           END IF
6240  !           LET Dsplc(Xcnt,Ycnt)=Dsplc(Xcnt,Ycnt+1)! Displacement mag.
6250  !         END IF
6260  !       NEXT Nd
6270  !     NEXT Ycnt
6280  !   END IF
6290  !   MAT Potnt= Potnt*(Resp)     ! Normalizing to 1 volt across sample
6300  !   MAT Dsplc= Dsplc*(Resp)     ! & sum of displacements along row=diel
6310  !   LET Nd1=1                   ! sign provider for following loop
6320  !   FOR Xcnt=1 TO Lside
6330  !     LET Resp2=Resp2+Nd1*(Dsplc(Xcnt,Boxes)-Dsplc(Xcnt,Boxes+1))
6340  !     LET Nd1=-Nd1
6350  !   NEXT Xcnt
6360  !   LET Resp2=Resp2/2           ! dielectric response perp to E
6370  !***> Should be end of calculations, printouts follow
6380  !***> Printout of the dielectric pixel array
6390   IF NOT (Sprss) AND Lside<10 THEN
6400     PRINT "DIELECTRIC PIXEL ARRAY, 3-dimensional,";Lside;"x";Lside;"x";Lside
6410     FOR Xcnt=1 TO Lside
6420       PRINT RPT$(" ",Lside);"X plane=";VAL$(Xcnt)
6430       FOR Zcnt=Lside TO 1 STEP -1
6440         FOR Ycnt=1 TO Lside
6450           PRINT "(";VAL$(DROUND(REAL(Diel(Pixl(Xcnt,Ycnt,Zcnt))),3));",";
6460           PRINT VAL$(DROUND(IMAG(Diel(Pixl(Xcnt,Ycnt,Zcnt))),3));") ";
```

```
6470          NEXT Ycnt
6480          PRINT
6490        NEXT Zcnt
6500      NEXT Xcnt
6510    END IF
6520   !***> Printout of the hopper array
6530    !****>  Find the series<->parallel factor
6540    LET Ctmp=FNWnr(Diel(*),Frpx(*),Resp,Tmp,9)
6550    PRINT
6560    PRINT "Composite Dielectric Response Tensor Components:"
6570    PRINT " principal=";Resp
6580    PRINT "  & series<->parallel factor =";
6590    PRINT "(";VAL$(DROUND(REAL(Ctmp),4));",";VAL$(DROUND(IMAG(Ctmp),4));")";
6600    PRINT "(+/- ";VAL$(DROUND(100*Tmp,4));"% iteration error)"
6610    PRINT
6620   !  IF Sprss=0 THEN
6630   !    PRINT "PIXEL VOLTAGES 2-dimensional,";Lside;"by";Lside
6640   !    Tmp=FNMatprnt(Potnt(*),-Lside)
6650   !    PRINT "PIXEL DISPLACEMENT FIELD MAGNITIUDES,";Lside;"by";Lside
6660   !    Tmp=FNMatprnt(Dsplc(*),-Lside)
6670   !  END IF
6680      !***> NOTE: Tranpose used then it is an additional cycle to Rptr
6690   !Graf(Grpt,1)=Rptr+Tmp/100
6700   !Graf(Grpt,2)=Frpx(1)
6710   !Graf(Grpt,3)=REAL(Ctmp)          ! save real part of exp ave factor
6720   !Graf(Grpt,4)=IMAG(Ctmp)          ! save imag part of exp ave factor
6730    Graf(Grpt,1)=(Rptr+Tmp/100)*.25+Graf(Grpt,1)+.125
6740    Graf(Grpt,2)=Frpx(1)*.5+Graf(Grpt,2)
6750    Graf(Grpt,3)=REAL(Ctmp)*.5+Graf(Grpt,3)
6760    Graf(Grpt,4)=IMAG(Ctmp)*.5+Graf(Grpt,4)
6770   !NEXT Trans
6780  NEXT Rptr
6790  !***> output repeat calculations
6800  LET Stmrp=TIMEDATE-Stmrp            ! Repeat time elapsed
6810  IF Stmrp>300 THEN BEEP              ! Beep if longer than 5 minutes
6820  IF Rptr>1 THEN PRINT "Finished";Rptr-1;"repeat trials in";Stmrp;"seconds."
6830  LET Bhdr$="("&VAL$(Lside)&"x"&VAL$(Lside)&"x"&VAL$(Lside)&")"
6840  IF Tls=1 THEN LET Bhdr$=Bhdr$&" elmnts"
6850  IF Tls=2 THEN LET Bhdr$=Bhdr$&"/(2x2x2s)"
6860  IF Kond=1 THEN LET Bhdr$=Bhdr$&" InslBC"
6870  IF Kond=2 THEN LET Bhdr$=Bhdr$&" PrdcBC"
6880  LET Bhdr$=Bhdr$&" Sparse"     ! solution by sparse methods
6890  IF Qdrnt=2 THEN LET Bhdr$=Bhdr$&" 4fold"
6900  IF Ptrn=0 THEN Bhdr$=Bhdr$&" intrnl,"
6910  IF Ptrn=1 THEN Bhdr$=Bhdr$&" "&Fln$
6920  IF Ptrn=2 THEN Bhdr$=Bhdr$&" USER,"
6930  IF Ptrn=3 THEN Bhdr$=Bhdr$&" RANDOM,"
6940  IF Ptrn=4 THEN Bhdr$=Bhdr$&" SLANT,"
6950  IF Ptrn=5 THEN Bhdr$=Bhdr$&" ELLIPSE,"
6960  IF Ptrn=6 THEN Bhdr$=Bhdr$&" STRAT,"
6970  IF Ptrn=7 THEN Bhdr$=Bhdr$&" BOXES,"
6980  LET Occp=LEN(Bhdr$)
6990  LET Bhdr$[1+Occp]=RPT$(" ",80-Occp)    ! pad with blanks
7000  LET Bhdr$[60]=" "&DATE$(TIMEDATE)&","&TIME$(TIMEDATE)
7010  LET Dhdr$=Bhdr$
```

```
7020  IF Rep=1 THFN PRINT "  for the case abbreviated .."
7030  IF Rep=1 THEN PRINT Bhdr$
7040  IF Rep>1 THEN
7050    PRINT " Summary of";Grpt;"repeat variations: (as programmed)"
7060    FOR Rptr=1 TO Grpt
7070      PRINT " Case";((Rptr-1) DIV Pose)+1;")",DROUND(Graf(Rptr,1),4),
7080      PRINT DROUND(Graf(Rptr,2),4),DROUND(Graf(Rptr,3),4),
7090      PRINT DROUND(Graf(Rptr,4),4)
7100    NEXT Rptr
7110    IF Nsvf=0 THEN
7120      DISP " Save repeat info (array form,";SIZE(Graf,1);"x";SIZE(Graf,2);
7130      INPUT ")? 0) No 1) Definitely",Nd1
7140    ELSE
7150      LET Nd1=0
7160      IF Fsv$<>"" THEN
7170        REDIM Graf(1:Grpt,1:Kwd)
7180        CREATE Fsv$,1 !<--<DOS if 1 unit
7190        ASSIGN @Savstr TO Fsv$;FORMAT OFF
7200        OUTPUT @Savstr;Ahdr$,Bhdr$,Grpt,Kwd,Graf(*),END
7210        ASSIGN @Savstr TO *
7220      ELSE
7230        Nd1=1
7240      END IF
7250    END IF
7260    IF Nd1=1 THEN
7270      DISP " Enclose (in """"s) new file name to send info vectors to?";
7280      INPUT " (@/null=use old file)",Fln$
7290      IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
7300      LINPUT " Title, (up to 80 characters)",Ahdr$
7310      LET Ahdr$[1+LEN(Ahdr$)]=RPT$(" ",80-LEN(Ahdr$))! pad with blanks
7320      DISP " File named """;Fln$;""" ([";LEN(Fln$);"] characters)";
7330      DISP " to contain repeat info"
7340      PRINT " File """;Fln$;"""'s user and description headers are ";
7350      PRINT "(2 lines):"
7360      PRINT Ahdr$
7370      PRINT Bhdr$
7380      IF Fln$="" OR Nsvf=1 THEN
7390        INPUT "  Enter the filename to be created? null=stop",Fln$
7400        IF Fln$="" THEN STOP
7410        !DISP " Enter file""";Fln$;"""'s storage size limit in bytes (`";
7420        DISP VAL$(256+8*Rep*Pose*Kwd);")";
7430        INPUT "?",Nd1
7440 !!     IF Nd1<1048 THEN Nd1=1048     ! 1 kiloBYTE min (LIF disks)
7450        CREATE Fln$,Nd1
7460      ELSE
7470        IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
7480      END IF
7490      ASSIGN @Infostr TO Fln$;FORMAT OFF
7500      OUTPUT @Infostr;Ahdr$,Bhdr$,Rep,Kwd,Graf(*),END
7510      ASSIGN @Infostr TO *
7520    END IF
7530  END IF
7540  !***> Pixel file output choice
7550  LET Nd1=0
7560 !IF Ptrn<>1 THEN INPUT " Save last pixel grid? 0)No 1)Yes",Nd1
```

```
7570 LET Nd1=0 !!temporary unitl reprogramming done
7580 IF Nd1=1 THEN
7590   DISP " Enclose (in """"s) new file name to send pixel pattern to?";
7600   INPUT " (null=use old file)",Fln$
7610   IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
7620   LINPUT " Title (up to 80 characters) if null then default label",Chdr$
7630   LET Dhdr$=Bhdr$
7640   DISP " File named """;Fln$;""" (["; LEN(Fln$);"] characters)";
7650   DISP " contains the pixel grid"
7660   PRINT " File """;Fln$;"""'s header is "
7670   PRINT Hdr$
7680   IF Fln$<>"" THEN
7690     DISP " Give file""";Fln$;"""'s max capacity limit in bytes";
7700     DISP "? (~";VAL$(128+SHIFT(Px_tot,-1));")";
7710     INPUT " ",Nd1
7720     IF Nd1<256 THEN Nd1=256
7730     CREATE Fln$,Nd1
7740   ELSE
7750     INPUT "  Enter the existing filename?",Fln$
7760     IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
7770   END IF
7780   ASSIGN @Pixstr TO Fln$;FORMAT OFF
7790   OUTPUT @Pixstr;Hdr$,Lside,Pixl(*),END
7800   ASSIGN @Pixstr TO *
7810 END IF
7820 !***> Interaction file output choice
7830 PRINT RPT$(" ",25);"...elapsed";PROUND(TIMEDATE-Start,-1);
7840 PRINT "sec for completion at ";TIME$(TIMEDATE)
7850 PRINT " MEMORY IS";VAL(SYSTEM$("AVAILABLE MEMORY"))/8;"(reals)"
7860 LET Lxtnt=Lside                     ! update COM /Pixel/ ie Pixl(*) size
7870 END
7880 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
7890 SUB Cv3ndi(INTEGER Nodi,Lszi,Sctri,Xouti,Youti,Zouti)
7900 ! Converts a node number of layering scheme into the (x,y,z)
7910 ! coordinates of neighboring nodes
7920 ! IN   Nodi = node number
7930 ! IN   Lszi = Pixl extent along either X or Y
7940 ! IN   Sctri = selection adjacent Pixl neighbor to node
7950 !      ie, quadrant number from binary (x/0/1 y/0/1 z/0/1)
7960 !      example 5>=>101 or x=1 y=0 z=1 or
7970 !      (0= X lower,Y lower Z lower; 1= X lower,Y lower, Z higher
7980 !       2= X lower, Y higher, Z lower; etc
7990 ! OUT Xouti = X coordinate address outcome
8000 ! OUT Youti = Y coordinate address outcome
8010 ! OUT Zouti = Z coordinate address outcome
8020 ! internal variables:
8030 !  Lyri = a counter for number of layers
8040 !  Swi = an even odd switch
8050 !  Ovri = overfill counter
8060 !  Nodmaxi = maximum node for given Pixel grid size, Lszi
8070 !  Hfszi = half of Pixel grid size, Lszi (then Lszi must be even)
8080 !  Sqvi = 1/4 of square whose edges are: Lszi*Lszi
8090 !  Bilv = the number of nodes between bilayers z={1,2 3,4 5,6 ..
8100   INTEGER Lyri,Swi,Ovri,Nodmaxi,Hfszi,Sqvi,Bilv
8110   LET Sctri=BINAND(Sctri,7)         ! mask only relevant bits
```

```
8120   IF BIT(Lszi,0) THEN PRINT " warning, odd Pixel grid extent"
8130   LET Sqvi=SHIFT(Lszi*Lszi,2)       ! 1/4 of square Lszi*Lszi
8140   LET Bilv=SHIFT(Lszi*Lszi,1)+Lszi-3 !Nodes in bilayer
8150   LET Nodmaxi=Sqvi*(Lszi+1)-SHIFT(5*Lszi,1)+4 !max node #
8160   LET Hfszi=SHIFT(Lszi,1)           ! effectively DIV 2 operation
8170   SELECT Nodi
8180   CASE <1                           ! grounded
8190     LET Xouti=Hfszi
8200     LET Youti=Hfszi
8210     LET Zouti=0
8220   CASE >=Nodmaxi                    ! exciter node
8230     LET Zouti=Lszi+1
8240     LET Youti=Hfszi
8250     LET Xouti=Hfszi
8260   CASE ELSE                         ! node is in cube
8270     LET Lyri=(Nodi-1) DIV Bilv            ! for Bilayer
8280     LET Ovri=(Nodi-1) MOD Bilv            ! for # Nodes in Bilayer itself
8290     LET Swi=(Ovri>=Sqvi)                  ! 0=lower 1=upper in Bilayer
8300     IF Swi THEN LET Ovri=Ovri-Sqvi        ! adjust for # Nodes in upper
8310     LET Zouti=1+SHIFT(Lyri,-1)+Swi+BIT(Sctri,0)
8320     IF Swi=0 THEN                         ! compute according to z level
8330       LET Youti=1+SHIFT(Ovri DIV Hfszi,-1)+BIT(Sctri,1)
8340       LET Xouti=1+SHIFT(Ovri MOD Hfszi,-1)+BIT(Sctri,2)
8350     ELSE
8360       IF Ovri<(Hfszi-1) THEN              ! along Y=1 edge
8370         LET Youti=BIT(Sctri,1)
8380         LET Xouti=SHIFT(Ovri,-1)+2+BIT(Sctri,2)
8390       ELSE                                ! not along Y=1 edge
8400         LET Ovri=Ovri-Hfszi+1
8410         LET Youti=Ovri DIV (Hfszi+1)
8420         LET Xouti=Ovri MOD (Hfszi+1)
8430         IF Youti=Hfszi-1 THEN             ! along Y=Lszi edge
8440           LET Xouti=SHIFT(Xouti,-1)+2+BIT(Sctri,2)
8450         ELSE                              ! somewhere in cube
8460           LET Xouti=SHIFT(Xouti,-1)+BIT(Sctri,2)
8470         END IF
8480         LET Youti=SHIFT(Youti,-1)+2+BIT(Sctri,1)
8490       END IF
8500     END IF
8510   END SELECT
8520 SUBEND
8530 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
8540 DEF FNNni(INTEGER Xn,Yn,Zn,Lszn,Lup)
8550 ! Returns the node number for Pixel grid network cube
8560 ! of capacitors for case of insulated sides.
8570 ! (Xn,Yn,Zn) 3D coordinates of Pixel leading to the nearest node
8580 ! If Up=1 then Pixl situated above node, else Pixl situated below node
8590 ! Where up is orientation towards exciter electrode
8600 !  & down is orientation towards grounded electrode
8610   INTEGER Ysw,Ndb,Lyr,Lsqhv,Ztyp,Zz
8620   LET Zz=Zn                        ! Copy of Z value for function call
8630   LET Lup=BIT(Lup,0)
8640   LET Lyr=SHIFT(Lszn,1)            ! in essence divides by 2
8650   LET Lsqv=SHIFT(Lszn*Lszn,2)      ! 1/4th of square Lszn*Lszn
8660   IF Xn>0 AND Xn<=Lszn AND Yn>0 AND Yn<=Lszn AND Zz>0 AND Zz<=Lszn THEN
```

```
8670      LET Ndb=1
8680      IF Zz=1 AND Lup THEN LET Ndb=0 !at ground electrode or next exciter
8690      IF Zz=Lszn AND NOT (Lup) THEN LET Ndb=Lsqv*(Lszn+1)-SHIFT(5*Lszn,1)+4
8700      REM above program line contains computation for EXCITER node number
8710      IF Ndb=1 THEN ! Node Numbering. Executes if node not yet assigned
8720        IF Zz<0 THEN Zz=-Zz           ! also entry line if node recalc
8730        LET Ztyp=BIT(Zz+1-Lup,0)      ! 0=no nodes on edge 1=on edge
8740        LET Ndb=(Zz-1-Lup)*Lsqv+SHIFT(Zz-Lup,1)*(Lszn-3)+Ztyp*(2-Lszn)
8750        LET Ndb=Ndb+(Lyr+Ztyp)*SHIFT(Yn+Ztyp-1,1) !node # up to y row
8760        LET Ndb=Ndb+SHIFT(Xn+1-Ztyp,1) ! node # up to x location
8770        IF Ztyp=1 THEN                ! special cases
8780          IF Yn=1 THEN
8790            LET Ndb=Ndb+1
8800            IF Xn=1 OR Xn=Lszn THEN Zz=-(Zz+(1-SHIFT(Lup,-1)))
8810          END IF
8820          IF Yn=Lszn THEN
8830            LET Ndb=Ndb-1
8840            IF Xn=1 OR Xn=Lszn THEN Zz=-(Zz+(1-SHIFT(Lup,-1)))
8850          END IF
8860        END IF !for(Ztyp=1)
8870      END IF !for(if Ndb=1)
8880      IF Zz<0 THEN GOTO 8720        !recalculate
8890    ELSE
8900      LET Ndb=-1
8910    END IF
8920    RETURN Ndb
8930  FNEND
8940  ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
8950  SUB Pixl3d_rand
8960  !***> Subprogram to create a 3D pixel array of random distribution
8970  !  @$%^%*^&%)!  R A N D O M   3 D   P I X E L   G R I D  &^#!#%)(%^$*
8980    COM /Pass/Relay(0:7)
8990    COM /Pixel/Chdr$[80],Dhdr$[80],INTEGER Lpix,Pixl(1:20,1:20,1:20)
9000    INTEGER Xp,Yp,Zp,Xq,Yq,Zq,Fill,Frdm,Sqrs,Pxs,Pixtmp,When,Lt,Ldb
9010    INTEGER Knpx,Nrdm,Rot,Dmn
9020   !LET Xq=SIZE(Pixl,1)<>Lpix OR SIZE(Pixl,2)<>Lside OR SIZE(Pixl,3)<>Lpix
9030    REDIM Pixl(1:Lpix,1:Lpix,1:Lpix) !? needed in sub"
9040    PRINT " Enjoy creating a randomized 3D pixel grid whose pixel elements"
9050    PRINT " are labelled 1..9"
9060    LET Nrdm=INT(Relay(0)+.0000001)    !Integer of relay
9070    IF Nrdm=0 THEN                      ! Manual seed
9080      INPUT "Random seed? (integer or neg if to be via timer)",Nrdm
9090      INPUT "Apply to 0) full 3D pixel cube 1) to 2D X-section ",Dmn
9100      IF Dmn>0 THEN
9110        INPUT "Contortions? none=0 compress=.5 elongate=2 ",Vchk
9120        IF Vchk>.25 AND Vchk<.75 THEN Vchk=1
9130        LET Lt=INT(Vchk+.5)
9140      ELSE
9150        LET Lt=0
9160      END IF
9170    ELSE
9180      LET Lt=INT(Relay(2)) MOD 3        ! Test of relay fraction 0,1,2
9190      LET Dmn=INT(.0000001+Relay(4))    ! Relay(4) pony for 2D or 3D
9200      IF Dmn=0 THEN Lt=0
9210    END IF
```

```
9220    LET Ldb=1+(Lt>0)                    ! Doubling factor
9230    IF Nrdm=0 THEN RANDOMIZE INT(TIMEDATE MOD 32767)
9240    IF Nrdm>0 THEN RANDOMIZE Nrdm
9250    LET Fill=0
9260    LET Sqrs=Lpix*Lpix/Ldb
9270    LET Knpx=Sqrs
9280    IF Dmn=0 THEN Knpx=Knpx*Lpix
9290    LET When=0
9300    LET Pxs=0
9310    LET Rq=0
9320    LET Nrdm=SHIFT(INT(Relay(0)+1.000001),1) !!Divide by 2 conj pairing
9330    IF Relay(0)>=1 AND Relay(1)<>0 THEN LET Rq=100*FRACT(Nrdm/Relay(1))
9340    WHILE Fill<Knpx
9350      LET Pxs=Pxs-1                     !Decrement
9360      WHILE Pxs<0
9370        LET When=When+1
9380        IF INT(Relay(0))=0 THEN
9390          DISP "Filling with component [";VAL$(When);"], give volume ";
9400          INPUT "percent (volume fraction * 100) ",Rq
9410        ELSE
9420          IF When>1 THEN Rq=-1          !Fill with rest, after 1st pass
9430        END IF
9440        IF Rq<0 THEN
9450          LET Pxs=Knpx-Fill
9460        ELSE
9470          LET Pxs=INT(Rq*Knpx*.01+.5) MOD Knpx
9480          IF Pxs>Knpx-Fill AND Knpx>Fill THEN Pxs=Knpx-Fill
9490        END IF
9500        PRINT " Component [";VAL$(When);"] is assigned";Pxs*Ldb;"pixels"
9510      END WHILE
9520      IF Pxs>0 AND Dmn>0 THEN
9530        LET Fill=Fill+1                 !Update current square location
9540        LET Zp=((Fill-1) DIV Lpix)+1    !Row of location
9550        LET Yp=Fill-(Zp-1)*Lpix         !Column of location
9560        LET Pixl(1,Zp,Yp)=When          !Assign pixel
9570      END IF
9580      IF Pxs>0 AND Dmn=0 THEN
9590        LET Fill=Fill+1                 !Update current square location
9600        LET Zp=((Fill-1) DIV Sqrs)+1    !Level between electrodes
9610        LET Xq=(Fill-1) MOD Sqrs        !remainder
9620        LET Yp=(Xq DIV Lpix)+1          !Y row in a Z level
9630        LET Xp=(Xq MOD Lpix)+1
9640        LET Pixl(Xp,Yp,Zp)=When         !X location in Y row
9650      END IF
9660    END WHILE
9670  !***> lotto-ing or random mixing
9680    IF Dmn=0 THEN
9690      FOR Fill=1 TO Knpx
9700        LET Frdm=INT(1+RND*Knpx)
9710        IF Fill<>Frdm AND Frdm<=Knpx THEN
9720          LET Zp=((Fill-1) DIV Sqrs)+1
9730          LET Xp=(Fill-1) MOD Sqrs
9740          LET Yp=(Xp DIV Lpix)+1
9750          LET Xp=(Xp MOD Lpix)+1
9760          LET Zq=((Frdm-1) DIV Sqrs)+1
```

```
9770          LET Xq=(Frdm-1) MOD Sqrs
9780          LET Yq=(Xq DIV Lpix)+1
9790          LET Xq=(Xq MOD Lpix)+1
9800          LET Pixtmp=Pixl(Xp,Yp,Zp)
9810          LET Pixl(Xp,Yp,Zp)=Pixl(Xq,Yq,Zq)
9820          LET Pixl(Xq,Yq,Zq)=Pixtmp
9830         !PRINT "r";VAL$(Frdm);"p";VAL$(Pixtmp); !check random sequencing
9840        END IF
9850      NEXT Fill
9860    ELSE
9870      FOR Fill=1 TO Sqrs
9880        LET Frdm=INT(1+RND*Sqrs)
9890        IF Fill<>Frdm AND Frdm<=Sqrs THEN
9900          LET Xp=1
9910          LET Zp=1+((Fill-1) MOD Lpix)
9920          LET Yp=1+((Fill-1) DIV Lpix)
9930          LET Xq=1
9940          LET Zq=1+((Frdm-1) MOD Lpix)
9950          LET Yq=1+((Frdm-1) DIV Lpix)
9960          LET Pixtmp=Pixl(Xp,Yp,Zp)
9970          LET Pixl(Xp,Yp,Zp)=Pixl(Xq,Yq,Zq)
9980          LET Pixl(Xq,Yq,Zq)=Pixtmp
9990         !PRINT "r";VAL$(Frdm);"p";VAL$(Pixtmp); !check random sequencing
10000       END IF
10010     NEXT Fill
10020     FOR Yp=1 TO Lpix             !swap
10030       FOR Zp=(1+Yp) TO Lpix
10040         LET Pixtmp=Pixl(1,Yp,Zp)
10050         LET Pixl(1,Yp,Zp)=Pixl(1,Zp,Yp)
10060         LET Pixl(1,Zp,Yp)=Pixtmp
10070       NEXT Zp
10080     NEXT Yp
10090     IF Ldb>1 THEN
10100       FOR Yp=Lpix TO 1 STEP -1    !Contort expand
10110         LET Yq=SHIFT(Xp+1,1)
10120         FOR Zp=1 TO Lpix
10130           LET Pixl(1,Yp,Zp)=Pixl(1,Yq,Zp)
10140         NEXT Zp
10150       NEXT Yp
10160     END IF
10170     FOR Xp=2 TO Lpix             !Copy over X planes
10180       FOR Yp=1 TO Lpix
10190         FOR Zp=1 TO Lpix
10200           LET Pixl(Xp,Yp,Zp)=Pixl(1,Yp,Zp)
10210         NEXT Zp
10220       NEXT Yp
10230     NEXT Xp
10240     IF Relay(0)=0 THEN INPUT "Rotate about Z axis? 0=N/1=Y ",Rot
10250     IF Rot=1 OR Relay(3)=1 THEN
10260       FOR Zp=1 TO Lpix             !swap
10270         FOR Yp=1 TO Lpix
10280           FOR Xp=(1+Yp) TO Lpix
10290             LET Pixtmp=Pixl(Xp,Yp,Zp)
10300             LET Pixl(Xp,Yp,Zp)=Pixl(Yp,Xp,Zp)
10310             LET Pixl(Yp,Xp,Zp)=Pixtmp
```

```
10320            NEXT Xp
10330          NEXT Yp
10340        NEXT Zp
10350      END IF
10360    END IF                          ! Dmn=0 for 3D or Dmn>1 for 2D
10370 SUBEND
10380 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
10390 DEF FNWnr(COMPLEX Diel(*),REAL Frpx(*),COMPLEX Din,REAL Alferr,INTEGER Nth)
10400   REM  Object of this function subprogram is to
10410   REM  find the exponential averaging factor (or
10420   REM  percolation related factor) "alf"
10430   REM  from a given set of complex number
10440   REM  dielectric values & fractional volume
10450   REM  weights and effective or resultant
10460   REM  complex dielectric value of composite
10470   REM           written by S. Wallin, 4/91.
10480   REM  The Wiener or exponential averaging factor
10490   REM  is defined as follows:
10500   REM  Diel0(resultant)^Alf = sum {Frpx(k)*Diel(k)^Alf}
10510   REM   where Diel0(resultant) = response of composite
10520   REM         Alf = exponential ave or Wiener or percolation factor
10530   REM         Frpx(k) = fractional volumes of species k
10540   REM         Diel(k) = (dielectric) response of species k
10550   COM /Pass/Relay(0:7)
10560   INTEGER I,J,K,K1,K2,Kdo,Ns,Lsn,Lst,New
10570   COMPLEX Diel0,Dlog0,Alf,Alf0,C0,C1,C2,C3,Clg,Clg2
10580   LET Ns=Nth
10590   IF Ns<=0 THEN STOP
10600   ALLOCATE COMPLEX Dlog(Ns)
10610   LET Avg=0
10620   LET Diel0=Din
10630   FOR I=1 TO Ns
10640     IF Diel(I)<>CMPLX(0,0) THEN Avg=Avg+Frpx(I)
10650   NEXT I
10660   REM Normalize ACTIVE volume to total 1
10670   FOR I=1 TO Ns
10680     !IF Avg<>0 THEN LET Frpx(I)=Frpx(I)/Avg
10690     IF Diel(I)=CMPLX(0,0) THEN Frpx(I)=0
10700   NEXT I
10710 !PRINT " Species data: (trial#, complex ";CHR$(238);" pair, adj vol wt)"
10720 !FOR I=1 TO Ns
10730  !PRINT " {#";I;"} (";REAL(Diel(I));",";IMAG(Diel(I));")",DROUND(Frpx(I),4)
10740 !NEXT I
10750 !PRINT " { eff} (";REAL(Diel0);",";IMAG(Diel0);")",1
10760   REM Determination of slope direction by log wt
10770   LET Dlog0=CMPLX(0,0)
10780   IF Diel0<>CMPLX(0,0) THEN Dlog0=LOG(Diel0)
10790   LET Clg=CMPLX(0,0)! Clg=Logarthimic mean
10800   LET Clg2=CMPLX(0,0)
10810   FOR I=1 TO Ns
10820     LET Dlog(I)=CMPLX(0,0)
10830     IF Diel(I)<>CMPLX(0,0) THEN Dlog(I)=LOG(Diel(I))-Dlog0
10840     LET Clg=Clg+Frpx(I)*Dlog(I)
10850     LET Clg2=Clg2+Frpx(I)*Dlog(I)*Dlog(I)
10860   NEXT I
```

```
10870   LET Lsn=-SGN(REAL(Clg))
10880 !PRINT " The logarithmic slope is ";DROUND(REAL(Clg),4);
10890 !PRINT DROUND(IMAG(Clg),4);"indicates ";CHR$(224);" is ";
10900 !IF Lsn=1 THEN !PRINT "positive."
10910 !IF Lsn=0 THEN !PRINT "at zero."
10920 !IF Lsn=-1 THEN !PRINT "negative."
10930   REM Extrema values
10940   LET Lst=0
10950   LET Zst=0
10960   FOR I=1 TO Ns
10970     IF Frpx(I)<>0 THEN
10980       LET Tmp=Lsn*ABS(Diel(I))
10990       IF Tmp>Zst OR Zst=0 THEN
11000         LET Zst=Tmp
11010         LET Lst=I
11020       END IF
11030     END IF
11040   NEXT I
11050   IF Lsn=0 THEN Lst=0
11060   LET Alf=CMPLX(0,0)
11070   IF Lst>0 AND Lst<=Ns+1 THEN
11080     IF Dlog(Lst)<>CMPLX(0,0) THEN Alf=-LOG(Frpx(Lst))/Dlog(Lst)
11090   END IF
11100   LET C0=CMPLX(Lsn,0)
11110   IF Clg2<>CMPLX(0,0) THEN C0=-2*Clg/Clg2!A 2nd guess
11120   LET Wt=ABS(C0)
11130   LET Wt=1/(1+Wt*Wt)!Relative weights for ave the 2 guesses
11140   LET Alf=Alf+Wt*(C0-Alf)!Combined 1st guess
11150  !PRINT "   Guess 1  ";CHR$(224);" = ";
11160  !PRINT DROUND(REAL(Alf),4);DROUND(IMAG(Alf),4)
11170   LET Alf0=CMPLX(0,0)
11180   LET Alferr=1
11190   LET J=2
11200   LET New=0
11210   WHILE Alf<>Alf0 AND J<32 AND Alferr>1.0E 13
11220     IF New=1 THEN
11230       LET Alf0=Alf!Keep track of last iteration
11240       New=0
11250     END IF
11260     LET C1=CMPLX(0,0)
11270     LET C2=CMPLX(0,0)
11280     LET C3=CMPLX(0,0)
11290     LET K=0!Keep count of non-zero terms
11300     FOR I=1 TO Ns
11310       LET C0=Alf*Dlog(I)
11320       IF ABS(REAL(C0))>700 THEN !Failure possible
11330         LET Alf=-2*Clg/Clg2
11340         LET Alf0=CMPLX(0,0)
11350         LET C0=CMPLX(0,0)
11360         LET C1=CMPLX(0,0)
11370         LET Alferr=0!Set to exit
11380       END IF
11390       IF C0<>CMPLX(0,0) AND Diel(I)<>0 THEN
11400         LET C0=Frpx(I)*EXP(C0)
11410         LET C1=C1+C0
```

```
11420          LET C2=C2+Dlog(I)*C0
11430          LET C3=C3+Dlog(I)*Dlog(I)*C0
11440          LET K=K+1!Tally another non-zero term
11450        END IF
11460      NEXT I
11470      IF C1<>CMPLX(0,0) AND K>1 THEN ! Log func deriv
11480        REM 0th, 1st, & 2nd logarithmic derivs
11490        LET C2=C2/C1
11500        LET C3=C3/C1-C2*C2
11510        LET C1=LOG(C1)
11520        REM Newton-Raphson estimate via 2nd degree polynomial
11530        LET K1=SGN(REAL(C1))
11540        LET K2=SGN(REAL(C2))
11550        LET C0=CMPLX(0,0)
11560        SELECT K2
11570        CASE 0
11580       !PRINT " o";
11590          IF C3<>CMPLX(0,0) THEN LET C0=-2*C1/C3
11600          IF C0<>CMPLX(0,0) THEN LET Alf=Alf+K1*SQR(C0)
11610        CASE Lsn
11620       !PRINT " +";
11630          LET C0=C2*C2-2*C1*C3
11640          IF C0=CMPLX(0,0) THEN
11650            LET C0=2*C1/C2
11660          ELSE
11670            LET C0=2*C1/(C2+K2*SQR(C0))
11680          END IF
11690          LET Alf=Alf-C0    !New estm of exp factor
11700        CASE -Lsn
11710       !PRINT " -";
11720          LET C0=C2*C2-2*C1*C3
11730          IF C0=CMPLX(0,0) THEN
11740            IF C3<>CMPLX(0,0) THEN LET C0=C2/C3
11750          ELSE
11760            IF C3<>CMPLX(0,0) THEN C0=(C2+K2*SQR(C0))/C3
11770          END IF
11780          LET Alf=Alf-C0    !New estm of exp factor
11790        END SELECT
11800        LET Alferr=ABS(REAL(Alf)-REAL(Alf0))+ABS(IMAG(Alf)-IMAG(Alf0))
11810        IF ABS(REAL(Alf))>700 THEN Alf=CMPLX(Lsn/2,0)-Clg/Clg2/(1+J)!Retry
11820        LET New=1                          !Set for update
11830      END IF
11840     !PRINT " Guess";J;" ";CHR$(224);" = ";Alf;" varied¯";Alferr
11850      LET J=J+1
11860    END WHILE
11870    LET Alferr=ABS(Alf0-Alf)    !Iteration variance
11880   !IF Alferr<>0 THEN !PRINT "  Iteration variance =";Alferr
11890    RETURN Alf
11900 FNEND
```

- * - * - * - * - * - * - * - * - * - * - * - * - * -

```
  - * - * - * - * - * - * - * - * - * - * - * - * - * -
                          "DIEL_SITES"
     Program in HP BASIC for numeric analysis based on the site model
  - * - * - * - * - * - * - * - * - * - * - * - * - * -

10     REAL Mem0,Tm0
20     LET Tm0=TIMEDATE
30     LET Mem0=INT(.5+VAL(SYSTEM$("AVAILABLE MEMORY")))
40     !   <   <   <   <   <   <     "DIEL_SITES"    >   >   >   >   >   >
50     !  * - * - * - * - * - * - * - * - * - * - * - * 
60     !   A main program to evaluate a box arrangement of pixel nodes     .
70     !   for the displacement fields and resultant dielectric response   .
80     !   of a composite where nodes and pixels are co-existent.          .
90     !                                         S. Wallin, June 1991      .
100    !  * - * - * - * - * - * - * - * - * - * - * - * 
110    CLEAR SCREEN
120    PRINT "Date: ";DATE$(Tm0);RPT$(" ",48);"Time: ";TIME$(Tm0)
130    PRINT " ";RPT$("  >  ",7);"""DIEL_SIM""";RPT$("  <  ",6);"  <"
140    PRINT RPT$(" ",56);"(S. Wallin, June 1991)"
150    PRINT
160    PRINT "     This program ""DIEL_SIM"" simulates a dielectric composite:";
170    PRINT " a box represented"
180    PRINT "as an interconnected electrical network.  The displacement field";
190    PRINT " (in analogy to "
200    PRINT "current) is a continuous quantity.  The electric field is the";
210    PRINT " voltage drop gra-"
220    PRINT "dient along the displacement field path segments.";
230    PRINT "  Schematically, each pixel is"
240    PRINT "cross joined between faces with a node."
250    PRINT "                                   . . . . ."
260    PRINT "            Paths                  .   |   .";
270    PRINT "         Node at"
280    PRINT "            connect                .___|___.";
290    PRINT "         center"
300    PRINT "            faces                  .   |   ."
310    PRINT "                                   .   |   ."
320    PRINT "                                   . . . . . ."
330    PRINT "The pixels are then interconnected into a nodal network."
340    PRINT "                                       |  <-< Exciter electrode"
350    PRINT "                                   ============="
360    WAIT .1
370    PRINT "                                    + + + + + +"
380    WAIT .1
390    PRINT "           X plane                  + + + + + +";
400    PRINT "         Y across"
410    WAIT .1
420    PRINT "           on page                  + + + + + +";
430    PRINT "         Z to/from"
440    WAIT .1
450    PRINT "           surface                  + + + + + +";
460    PRINT "         electrodes"
470    WAIT .1
480    PRINT "                                   ============="
```

```
490   WAIT .1
500   PRINT "                                     | <-< Ground electrode"
510   PRINT "Initial memory is";PROUND(Mem0/16,2);
520   PRINT "(available as complex number storage units)."
530   PRINT "[Hint: Use up/down arrow keys to scroll on CRT when paused";
540   PRINT " (halt=Clear"
550   PRINT "I/O & Stop, break=Pause & Stop).]"
560   ! * - * - * - * - * - * - * - * - * - * - * - *  !
570   !                        DATA                    !
580   ! Data is for use as default permittivities      !
590   DATA 1,0,2,2,4,6,5,8,6,16,7,32,8,64,9,128
600   ! * - * - * - * - * - * - * - * - * - * - * - *  !
610   !                    COMmon Memory               !
620   COM /Info/ INTEGER Dsrc,Kond,Ptrn,Knj,Spcs,Meth,Back,Sgj,Svr,Fln$[80],Msd$[60],COMPL
                  Dlct(0:9)
630   COM /Pass/Relay(0:7)
640   COM /Pixel/ INTEGER Xmax,Ymax,Zmax,Pixl(1:8000),REAL Xscl,Yscl,Zscl
650   COM /Memr/Graf(1:512,1:8),Ahdr$[80],Bhdr$[80],INTEGER Rep,Kwd
660   COM /Titles/Chdr$[80],Dhdr$[80]
670   ! * - * - * - * - * - * - * - * - * - * - * - *  !
680   !                      VARIABLES                 !
690   INTEGER Xadr,Yadr,Zadr,Xcnt,Ycnt,Zcnt
700   INTEGER Axs,Dtl,Face,Infc,Lpiv,Hdig,Hedge,Hlmt,Hmem,Hrel,Hrow,Hclm,Hcnt
710   INTEGER Nd1,Nd2,Ndmax,Nmr,Nwrk,Ovr,Rcyc,Rman,Rptr,Spc0,Zxy
720   REAL Iterr,Mem1,Ntrt,Norm,Tm1,Tmv0,Tmv1,Tmcyc,Tmup,Xadm,Yadm,Zadm,Frk(0:9)
730   COMPLEX Admt0,Admt1,Admt2,Alph,Hnrm,Resp
740   DIM Trs$[40],Ws$[250]
750   ! COM /Info/ variable descriptions:
760   !  Dsrc = Dielectric or permittivity selection
770   !  Kond = boundary condition 1) Insulative 2) Periodic or wrap around
780   !  Ptrn = Pixel fill pattern choice
790   !  Knj = permittivity conjugation, 0=none 1=alternating 2=averaged
800   !  Spcs = number of species or types
810   !  Meth = nodal analysis method 1=classic 2=sparseness advantage
820   !  Back = backsubtition indicator 0=off 1=on
830   !  Sgj =  conjugation state +1 or -1
840   !  Svr =  flag to save output on a file
850   !  Fln$ = name of file
860   !  Msd$ = name of mass storage device
870   !  Dlct(*) = array of complex dilectric values
880   ! COM /Pass/ Relay passes information to SUBprograms with 8 reals
890   ! COM /Pixel/ variable descriptions:
900   !  Xmax,Ymax,Zmax = greatest of each pixel address extent
910   !  Pixl(*) = integer array of pixel species for dielectric types
920   !  Xscl,Yscl,Zscl = real scaling factors
930   ! COM /Memr/ variable descriptions:
940   !  Graf(*) = a data output storage array kept after program run
950   !  Ahdr$ = 1st title line for output description
960   !  Bhdr$ = 2nd title line for output description
970   !  Rep = integer of number of repeats or length of array Graf(*)
980   !  Kwd = integer of width of array Graf(*)
990   ! Variables in the MAIN program:
1000  !  Xadr,Yadr,Zadr = integer pixel addresses in (X,Y,Z) coordinates
1010  !  Xcnt,Ycnt,Zcnt = pixel counter addresses in (X,Y,Z) coordinates
1020  !  Axs = 1,2,3 <=> X,Y,Z directions
```

```
1030  !  Dtl = 0 to surpress 1 to print, nodal analysis details
1040  !  Face = integer for 3D axis selection or sector
1050  !  Infc= interfaces tally
1060  !  Lpiv = pivoting occurances
1070  !  Hdig = a counter used for hopper/sparse nodal analysis
1080  !  Hedge = # along edge of hopper/sparse nodal analysis technique
1090  !  Hmem = hopper storage requirement
1100  !  Hrel,Hrow,Hclm,Hcnt = row & column addresses in hopper/sparse tech
1110  !  Nd1 = integer of primary node address
1120  !  Nd2 = integer of a node adjoining primary node
1130  !  Ndmax = node maximum = 1+Xmax*Ymax*Zmax
1140  !  Nwrk = work integer
1150  !  Ovr = integer indicating for overwrite/append output file
1160  !  Rcyc = integer flag to reuse COM /Info/ variables as input
1170  !  Rman = manual flag 0=OFF 1=ON
1180  !  Rptr = integer overall repeat counter
1190  !  Spc0 = integer used in determining species limit
1200  !  Zxy = integer number of pixels on a Z plane, Xmax*Ymax
1210  !  Ntrt = real number of integer value for sparseness memory array
1220  !  Norm = real normalization constant, Zmax/Xmax/Ymax
1230  !  Frk(*) = real array of volume fractions of different types
1240  !  Admt0,Admt1,Admt2 =  complex admittance paths
1250  !  Alph = series<=>parallel, Wiener, or exponential averaging factor
1260  !  Hnrm = normalizing element for hopper pivoting
1270  !  Resp = resultant permittivity
1280  !  Trs$,Ws$ = work string of 40 chars,250
1290  ! Variable arrays to be ALLOCATED
1300  !  Act1(*),Act2(*) = complex nodal interaction array
1310  !  Tca1(*),Tca2(*) = complex inverse of nodal interactoin array
1320  !  Ptn(*) = complex potentials (w/ backsubstition)
1330  !  Endex(*) = complex exciter interactions (w/ backsubstition)
1340  !  Iterr = real iteration error
1350  !  Mem1,Mem0 = real available internal memory for program
1360  !  Tm0,Tm1,Tmv0,Tmv1,Tmcyc,Tmup = real start and stop times
1370  !  Xadm,Yadm,Zadm = pixel face admittance factors
1380  IF Zmax>0 THEN                    ! presumably now a rerun
1390    LET Rcyc=0
1400    IF Kond>0 THEN
1410      INPUT "Rerun at previous settings? 0=no 1=yes (default=0) ",Rcyc
1420    END IF
1430    IF Rcyc<0 THEN STOP             !panic stop
1440    LET Rcyc=(Rcyc>0)
1450  END IF
1460  IF Rcyc THEN                      ! reuse previous inputs, Rcyc=1
1470    LET Zxy=Xmax*Ymax               ! Pixs on a Z level
1480    LET Ndmax=Zxy*Zmax+1            ! Exciter node
1490    LET Xadm=Yscl*Zscl/Xscl         ! X face admittance factor
1500    LET Yadm=Xscl*Zscl/Yscl         ! Y face admittance factor
1510    LET Zadm=Xscl*Yscl/Zscl         ! Z face admittance factor
1520    LET Norm=Zmax/Zadm/Zxy          ! Normalization for permittivity
1530    PRINT "From COM /Memr/ pixel array is";Xmax;"x";Ymax;"x ";VAL$(Zmax);"."
1540    IF Xmax<=0 OR Ymax<=0 OR Zmax<=0 THEN LET Rcyc=0
1550    LET Ws$=""
1560    FOR Xcnt=1 TO Spcs              ! set up a current title
1570      LET Ws$=Ws$&"{"&VAL$(Xcnt)&"}s"&"="
```

```
1580      LET Ws$=Ws$&"("&VAL$(DROUND(REAL(Dlct(Xcnt)),3))
1590      LET Ws$=Ws$&","&VAL$(DROUND(IMAG(Dlct(Xcnt)),3))&")"
1600    NEXT Xcnt
1610    LET Ycnt=MIN(LEN(Ws$),75)
1620    LET Ahdr$="Diels "&Ws$[1;Ycnt]
1630  ELSE                              ! get new input info, Rcyc=0
1640    LET Nwrk=0
1650    DISP "IO to be 0) default 1) lab 2) hardisk 3,4) office 5) user defined";
1660    INPUT " (default=0) ",Nwrk
1670    SELECT Nwrk
1680    CASE <0
1690      STOP
1700    CASE =0
1710      LET Msd$=""
1720    CASE =1
1730      LET Msd$=":CS80,700,1"
1740    CASE =2
1750      LET Msd$=":CS80,700,0"
1760    CASE =3
1770      LET Msd$=":CS80,700,0"
1780    CASE =4
1790      LET Msd$=":CS80,703,1"
1800    CASE =5
1810      LINPUT "Name your storage device_ ",Msd$
1820    CASE ELSE
1830      DISP "Mass storage selection too big for menu, defaults"
1840      LET Msd$=""
1850    END SELECT
1860    IF Msd$<>"" THEN PRINT RPT$(" ",48);"mass storage device ";Msd$
1870    PRINT "Indicate Pixel fill pattern:"
1880    PRINT "  0) internal, via COM /Memr/"
1890    PRINT "  1) from file storage"
1900    PRINT "  2) hand fill, (tedious)"
1910    PRINT "  3) random, basic shuffle of pixel iso-sized grains"
1920    PRINT "  4) random, shuffle with dual blocking"
1930    PRINT "  5) ellipsoid, (semi-model of effective medium theory)"
1940    PRINT "  6) correlation, (exponential, power laws)"
1950    PRINT "  7) fractal with evolution by successive generations"
1960    PRINT "     default is currently ";VAL$(Ptrn);"."
1970    LET Ndl=Ptrn                      ! keep track of previous
1980    IF Kond=0 THEN Ptrn=3
1990    DISP "Pixel fill selection? (default=";VAL$(Ptrn);
2000    INPUT ") ",Ptrn
2010    IF Ptrn<0 THEN STOP
2020    IF Ptrn<>Ndl THEN MAT Relay=(0)  ! zero upon Ptrn change
2030    SELECT Ptrn
2040    CASE =0
2050      IF Xmax<=0 OR Ymax<=0 OR Zmax<=0 THEN
2060        DISP "May have memory error, pixel box measures ";
2070        DISP VAL$(Xmax);"x";VAL$(Ymax);"x";VAL$(Zmax)
2080        STOP
2090      END IF
2100      PRINT "The pixels are from internal memory and measure ";
2110      PRINT VAL$(Xmax);"x";VAL$(Ymax);"x";VAL$(Zmax)
2120    CASE =1
```

```
2130      LINPUT "File source name for Pixels? ",Fln$
2140      IF Fln$="" THEN STOP
2150      IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
2160      DISP "File """;Fln$;""" is being read from storage"
2170      ASSIGN @Pxsrc TO Fln$;FORMAT OFF
2180      ENTER @Pxsrc;Ahdr$;Bhdr$;Xmax;Ymax;Zmax;Xscl;Yscl;Zscl
2190      IF Xmax<=0 OR Ymax<=0 OR Zmax<=0 THEN
2200        DISP """";Fln$;""" may have error, pixel box measures ";
2210        DISP VAL$(Xmax);"x";VAL$(Ymax),"x";VAL$(Zmax)
2220        ASSIGN @Pxsrc TO *
2230        STOP
2240      END IF
2250      IF Xmax<=0 OR Ymax<=0 OR Zmax<=0 THEN PRINT Xmax;"x";Ymax;"x";Zmax;"?"
2260      FOR Xcnt=1 TO Xmax*Ymax*Zmax
2270        ENTER @Pxsrc;Pixl(Xcnt)
2280      NEXT Xcnt
2290      ASSIGN @Pxsrc TO *
2300    END SELECT
2310    IF Ptrn>2 THEN                    !get Pixel limits from user
2320      IF Xmax<1 THEN Xmax=1
2330      DISP "Give X pixel limit, (how many X planes, default=";VAL$(Xmax);
2340      INPUT ")? ",Xmax
2350      IF Xmax<0 THEN STOP
2360      IF Xmax=0 THEN Xmax=1
2370      IF Ymax<1 THEN Ymax=1
2380      DISP "Give Y pixel limit, (how many Y columns/plane,";
2390      DISP " default=";VAL$(Ymax);
2400      INPUT ");? ",Ymax
2410      IF Ymax<0 THEN STOP
2420      IF Ymax=0 THEN Ymax=1
2430      IF Zmax<1 THEN Zmax=1
2440      DISP "Give Z pixel limit, (how many Z rows between electrodes,";
2450      DISP " default ";VAL$(Zm[illegible]);
2460      INPUT ")? ",Zmax
2470      IF Zmax<0 THEN STOP
2480      IF Zmax=0 THEN Zmax=1
2490      IF Xscl<=0 THEN Xscl=1
2500      DISP "Give X pixel scale factor (>0 to inf, default=";VAL$(Xscl);
2510      INPUT ")? ",Xscl
2520      IF Xscl<0 THEN STOP
2530      IF Xscl=0 THEN Xscl=1
2540      IF Yscl<=0 THEN Yscl=1
2550      DISP "Give Y pixel scale factor (>0 to inf, default=";VAL$(Yscl);
2560      INPUT ")? ",Yscl
2570      IF Yscl<0 THEN STOP
2580      IF Yscl=0 THEN Yscl=1
2590      IF Zscl<=0 THEN Zscl=1
2600      DISP "Give Z pixel scale factor (>0 to inf, default=";VAL$(Zscl);
2610      INPUT ")? ",Zscl
2620      IF Zscl<0 THEN STOP
2630      IF Zscl=0 THEN Zscl=1
2640    END IF                            !end if for Ptrn>2 test
2650    PRINT " The scale factors are: X's=";VAL$(DROUND(Xscl,4));
2660    PRINT ", Y's=";VAL$(DROUND(Yscl,4));", & Z's=";VAL$(DROUND(Zscl,4))
2670    IF Kond=0 THEN Kond=1
```

```
2680  DISP "Boundary conditions? 1) Insulated 2) Periodic or";
2690  DISP " wrap around (default=";VAL$(Kond);
2700  INPUT ") ",Kond
2710  IF Kond<0 THEN STOP
2720  LET Kond=(Kond>1)+1
2730  LET Spc0=9                    ! overall species limit set at 9
2740  IF Spcs=0 THEN Spcs=2         ! binary species default
2750  DISP "How many species overall? (<=";VAL$(Spc0);",default=";VAL$(Spcs);
2760  INPUT ")",Spcs                ! later.. incl option for subspecies
2770  IF Spcs<0 THEN STOP           ! panic stop
2780  IF Spcs=0 THEN Spcs=2         ! default to binary composite
2790  IF Spcs>Spc0 THEN Spcs=9       ! reset to programming limit
2800  IF Dlct(1)=CMPLX(0,0) THEN LET Dsrc=1 !default
2810  DISP "Permittivi`ies from? 0=intrnl mem 1=prog 2=user ";
2820  IF Spcs=2 THEN DISP "3=binary insl/cond ";
2830  DISP "(default=";VAL$(Dsrc);
2840  INPUT ")",Dsrc
2850  IF Dsrc<0 THEN STOP
2860  IF Dsrc>3 THEN Dsrc=2
2870  IF Dsrc>0 THEN LET Sgj=1       ! reset conjugation state to fresh=+1
2880  SELECT Dsrc
2890  CASE =1
2900    FOR Xcnt=1 TO Spcs
2910      READ Dlct(Xcnt)
2920    NEXT Xcnt
2930  CASE =2
2940    FOR Xcnt=1 TO Spcs
2950      DISP "Complex permittivity for species [";VAL$(Xcnt);
2960      DISP "]s (previous=(";VAL$(DROUND(REAL(Dlct(Xcnt)),4));",";
2970      DISP VAL$(DROUND(IMAG(Dlct(Xcnt)),4));"))";
2980      I[illegible]     ",Dlct(Xcnt)
2990    NEXT Xcnt
3000  CASE =3
3010    FOR Xcnt=1 TO Spcs
3020      IF BIT(Xcnt,0)=0 THEN
3030        LET Iterr=REAL(Dlct(Xcnt))
3040        DISP "For the species [";VAL$(Xcnt);"], give the real part?";
3050        DISP " (previous=";VAL$(DROUND(Iterr,4));
3060        INPUT ") ",Iterr
3070        LET Dlct(Xcnt)=CMPLX(Iterr,0)
3080      ELSE
3090        LET Iterr=IMAG(Dlct(Xcnt))
3100        DISP "For the species [";VAL$(Xcnt); ], give the imaginary";
3110        DISP " (loss) part? (previous=";VAL$(DROUND(Iterr,4));
3120        INPUT ") ",Iterr
3130        LET Dlct(Xcnt)=CMPLX(0,Iterr)
3140      END IF
3150    NEXT Xcnt
3160  END SELECT
3170  IF Meth=0 THEN Meth=2
3180  DISP "Select nodal analysis method 1) classic matrix";
3190  DISP " 2) sparse advantage (default=";VAL$(Meth);
3200  INPUT ") ",Meth
3210  IF Meth<0 THEN STOP            !panic stop
3220  LET Meth=1+(Meth>1)
```

```
3230    DISP "Backsubstitution, ie backchecking & field mapping? ";
3240    DISP "0=no 1=yes (default=";VAL$(Back);
3250    INPUT ")",Back
3260    IF Back<0 THEN STOP
3270    LET Back=(Back>0)
3280    DISP "How many repeats? (default=";VAL$(Rep);
3290    DISP ", 0=manual, ";VAL$(SIZE(Graf,1));"=max prog)";
3300    INPUT " ",Rep
3310    IF Rep<0 THEN STOP
3320    LET Ws$=""
3330    FOR Xcnt=1 TO Spcs              ! set up title
3340      LET Ws$=Ws$&"["&VAL$(Xcnt)&"]s"&"="
3350      LET Ws$=Ws$&"("&VAL$(DROUND(REAL(Dlct(Xcnt)),3))
3360      LET Ws$=Ws$&","&VAL$(DROUND(IMAG(Dlct(Xcnt)),3))&")"
3370    NEXT Xcnt
3380    LET Ycnt=MIN(LEN(Ws$),75)
3390    LET Ahdr$="Diels "&Ws$[1;Ycnt]
3400    IF Rep>1 THEN
3410      DISP "Conjugate pairing (by 2s)? 0=no 1=alternating 2=averaged ";
3420      DISP "(default=";VAL$(Knj);")";
3430      INPUT " ",Knj
3440      IF Knj<0 THEN STOP
3450      IF Knj>2 THEN Knj=0
3460      DISP "Save repeat information in a file?";
3470      DISP " 0=no 1=yes (default=";VAL$(Svr);
3480      INPUT ") ",Svr
3490      IF Svr<0 THEN STOP            ! panic stop
3500      LET Svr=(Svr>0)
3510      IF Svr THEN
3520        LET Ws$=Fln$                !for a suggested file naming
3530        IF Ptrn>0 THEN
3540          IF Ptrn=1 THEN Fln$="S"
3550          IF Ptrn=2 THEN Fln$="H"
3560          IF Ptrn=3 THEN Fln$="G"
3570          IF Ptrn=4 THEN Fln$="B"
3580          IF Ptrn=5 THEN Fln$="E"
3590          IF Ptrn=6 THEN Fln$="K"
3600          IF Ptrn=7 THEN Fln$="F"
3610          LET Xcnt=MIN(Xmax,Ymax,Zmax)
3620          LET Ycnt=MAX(Xmax,Ymax,Zmax)
3630          IF Xcnt=Ycnt AND Xmax>9 THEN
3640            Fln$=Fln$&"3D"&VAL$(Xmax)
3650          ELSE
3660            LET Fln$=Fln$&VAL$(Xmax)&VAL$(Ymax)&VAL$(Zmax)
3670          END IF
3680          IF Kond=1 THEN Fln$=Fln$&"S"
3690          IF Kond=2 THEN Fln$=Fln$&"P"
3700        END IF                      ! end if, new suggested filename
3710        IF Ptrn>2 THEN
3720          IF Dlct(2)<>CMPLX(0,0) THEN Admt1=LOG(Dlct(1)/Dlct(2))
3730          LET Admt1=CMPLX(REAL(Admt1)/LOG(10),IMAG(Admt1)*180/PI)*1000
3740          LET Trs$=VAL$(ABS(REAL(Admt1)))
3750          IF ABS(REAL(Admt1))<1000 THEN Trs$="0"&Trs$
3760          LET Fln$=Fln$&Trs$[1;2]
3770          LET Trs$=VAL$(ABS(IMAG(Admt1)))
```

```
3780          IF ABS(REAL(Admtl))<900 THEN Trs$="0"&Trs$
3790          LET Fln$=Fln$&Trs$[1;1]
3800        END IF
3810        OUTPUT KBD;Fln$;
3820        DISP "Verify or name file to receive output (change & enter";
3830        IF Ws$<>"" THEN DISP " or type old=""";Ws$;"""";
3840        LINPUT ") ",Fln$
3850        OUTPUT KBD;Ahdr$;
3860        DISP "Verify or enter a title? (<80";
3870        IF Ahdr$<>"" THEN DISP ", prev run=";Chdr$;
3880        LINPUT ") ",Ahdr$
3890      END IF                        ! end if for Svr
3900    ELSE
3910      LET Knj=0                     ! if no repeats, then no conjugation
3920    END IF                          ! end if for Rep>0
3930  END IF                            ! end if for Rcyc=0
3940  LET Rman=(Rep=0)                  ! manual switch
3950  IF Svr AND Fln$="" THEN Svr=0     ! if nothing change to NO filing
3960  IF Svr THEN                       ! anticipate save output to file
3970    IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$ !add mass storage specifier
3980    ON ERROR GOTO 4090              ! error means branch to new file
3990    ASSIGN @Exist TO Fln$
4000    ASSIGN @Exist TO *
4010    OFF ERROR
4020    LET Ovr=0
4030    DISP """";Fln$;""" exists.  Select: 0=skip filing 1=overwrite";
4040    INPUT " 2=append ",Ovr
4050    IF Ovr<0 THEN STOP
4060    IF Ovr>2 THEN Ovr=0
4070    IF Ovr=0 THEN Svr=0
4080    GOTO 4120                       ! branch to end file check
4090    LET Ovr=3                       ! create output file
4100    OFF ERROR
4110    ASSIGN @Exist TO *
4120    REM end of file existance checking
4130  END IF                            ! end if to Svr=save output to file
4140  PRINT "Selections: Pixel pattern =";Ptrn;"& Boundary condition =";Kond
4150  LET Zxy=Xmax*Ymax                 ! Max nodes residing on a Z level
4160  LET Ndmax=Zxy*Zmax+1              ! Exciter node number
4170  LET Xadm=Yscl*Zscl/Xscl           ! X face admittance factor
4180  LET Yadm=Xscl*Zscl/Yscl           ! Y face admittance factor
4190  LET Zadm=Xscl*Yscl/Zscl           ! Z face admittance factor
4200  LET Norm=Zmax/Zadm/Zxy            ! Normalization value
4210  PRINT "The box measures";Xmax;"x";Ymax;"x";Zmax;"pixels giving";
4220  PRINT Ndmax-1;"total."
4230  PRINT "There is a node for each pixel plus the exciter electrode, so";
4240  PRINT " in all ";VAL$(Ndmax);"."
4250  LET Dtl=0                         !Default to no detail
4260  IF Ndmax<37 AND Rman THEN
4270    DISP "View/check nodal ";VAL$(Ndmax);"x";VAL$(Ndmax);" analysis";
4280    INPUT " details? 0=no 1=yes (default=0) ",Dtl
4290    IF Dtl<0 THEN STOP
4300    LET Dtl=(Dtl>0)
4310  END IF                            ! end  if Rep=1 & Ndmax condition
4320  IF Dtl THEN
```

```
4330    PRINT "Nodal analysis numbering scheme"
4340    FOR Xcnt=1 TO Xmax
4350      PRINT RPT$(" ",SHIFT(Ymax,-1));"X=";VAL$(Xcnt)
4360      FOR Zcnt=Zmax TO 1 STEP -1
4370        FOR Ycnt=1 TO Ymax
4380          LET Nwrk=FNOde_mk(Xcnt,Ycnt,Zcnt,Xmax,Ymax,Zmax)
4390          PRINT USING "4D,#";Nwrk
4400          CALL Node_addrs(Nwrk,Xadr,Yadr,Zadr,Xmax,Ymax,Zmax)
4410          IF Xadr<>Xcnt OR Yadr<>Ycnt OR Zadr<>Zcnt THEN
4420            PRINT "(";
4430            IF Xcnt<>Xadr THEN PRINT "X";VAL$(Xcnt);"-";VAL$(Xadr);
4440            IF Ycnt<>Yadr THEN PRINT "Y";VAL$(Ycnt);"-";VAL$(Yadr);
4450            IF Zcnt<>Zadr THEN PRINT "Z";VAL$(Zcnt);"-";VAL$(Zadr);
4460            PRINT ")";
4470          END IF
4480        NEXT Ycnt
4490        PRINT
4500      NEXT Zcnt
4510    NEXT Xcnt
4520  END IF
4530  IF Meth=1 THEN                        !Initialize nodal analysis arrays
4540    ALLOCATE COMPLEX Act1(1:Ndmax,1:Ndmax),Tcal(1:Ndmax,1:Ndmax)
4550    ALLOCATE COMPLEX Ptn(1:Ndmax),Endex(1:Zxy+1),Fluz(1:Ndmax-1)
4560    LET Ntrt=2.0*Ndmax*Ndmax+Ndmax  !number of storage elements
4570  ELSE
4580    LET Hedge=Zxy+1                     !edge units along side of nodal
4590    LET Hmem=(1.0+Hedge)*Hedge*.5   !analysis hopper overall #=Hmem
4600    ALLOCATE COMPLEX Hpiv(1:Hedge),Act2(1:Hmem)
4610    LET Ntrt=(1.0+Ndmax)*Ndmax*.5   !Max nodal storage requirement
4620    ON ERROR GOTO 4670                  !Branch on error to NO backsub
4630    ALLOCATE COMPLEX Tca2(1:Hedge-1,1:Ndmax-1),Ptn(1:Ndmax)
4640    ALLOCATE COMPLEX Endex(1:Hedge),Fluz(1:Ndmax-1)
4650    GOTO 4710                           !else error occured
4660    IF Back=1 THEN
4670      PRINT "Memory too small to save for backsubtitution,";
4680      PRINT " thus NO BACKCHECKING or NO POTENTIALS."
4690    END IF
4700    LET Back=0                          !no backsubstituting
4710    OFF ERROR
4720  END IF
4730  !  =   =   =   =   N O D A L   A N A L Y S I S   =   =   =   =   =
4740  LET Kwd=SIZE(Graf,2)                   ! row size of Graf storage
4750  MAT Graf=(0)                           ! initialize Graf storage
4760  IF Ptrn=2 THEN MAT Relay=(0)          ! zero Relays for manual fill
4770  LET Tmcyc=TIMEDATE                     ! start cycle timer
4780  FOR Rptr=1 TO SHIFT(Rep,-(Knj>0))+Rman! repeat cycles, use conjugation
4790    IF Meth=1 THEN                       ! initialize nodal arrays to zero
4800      MAT Act1=(CMPLX(0,0))
4810      MAT Tcal=(CMPLX(0,0))
4820    ELSE
4830      MAT Act2=(CMPLX(0,0))
4840      IF Back THEN MAT Tca2=(CMPLX(0,0))
4850    END IF
4860  ! FILL PIXELS
4870    IF Ptrn>2 THEN MAT Pixl=(0)          ! initialize pixel array to zero
```

```
4880     LET Nd2=(Knj>0)                    ! use as a test of Knj in SELECT
4890     SELECT Ptrn
4900     CASE =2
4910       CALL Pixs_by_hand
4920     CASE =3
4930       IF NOT (Rcyc OR Rman) AND Rptr=1 THEN ! if multiple calls set Relay(2)
4940         LET Xadr=1
4950         INPUT "Shuffle along the X direction? 0=no 1=yes (default=1) ",Xadr
4960         IF Xadr<0 THEN STOP
4970         LET Xadr=(Xadr=0)
4980         LET Yadr=1
4990         INPUT "Shuffle along the Y direction? 0=no 1=yes (default=1) ",Yadr
5000         IF Yadr<0 THEN STOP
5010         LET Yadr=(Yadr=0)
5020         LET Zadr=1
5030         INPUT "Shuffle along the Z direction? 0=no 1=yes (default=1) ",Zadr
5040         IF Zadr<0 THEN STOP
5050         LET Zadr=(Zadr=0)
5060         REM Relay(0) = volume fractions
5070         REM Relay(1)=seed
5080         LET Relay(2)=SHIFT(Xadr,-2)+SHIFT(Yadr,-1)+Zadr
5090       END IF
5100       IF Rman THEN
5110         Relay(0)=0
5120         Relay(1)=0
5130       ELSE
5140         LET Relay(0)=FRACT(SHIFT(Rptr+Nd2,Nd2)/(Rep+1))!vol frac
5150         LET Relay(1)=Rptr              ! pass to randomize seed
5160       END IF
5170       CALL Pixs_by_random(Spcs)
5180     CASE =4
5190       REM Relay(0) = volume fractions
5200       REM Relay(1)=seed
5210       IF Rman THEN
5220         Relay(0)=0
5230         Relay(1)=0
5240       ELSE
5250         LET Relay(0)=FRACT(SHIFT(Rptr+Nd2,Nd2)/(Rep+1))!vol frac
5260         LET Relay(1)=Rptr              ! pass to randomize seed
5270       END IF
5280       CALL Pixs_by_2block(Spcs)
5290     CASE =5
5300       REM Relay(0) = volume fractions
5310       REM Relay(1,2,3)=ellipse axis
5320       REM Relay(4)
5330       IF NOT Rcyc AND Rptr=1 AND Rep>1 THEN
5340         DISP "Inclusion role? 0)=smaller species, 1)=species #1,";
5350         INPUT " 2)=species #2 ",Relay(4)
5360         IF Relay(4)<0 THEN STOP
5370         LET Relay(1)=1
5380         INPUT "Ellipsoid X axis length? (default=1) ",Relay(1)
5390         LET Relay(2)=1
5400         INPUT "Ellipsoid Y axis length? (default=1) ",Relay(2)
5410         LET Relay(3)=1
5420         INPUT "Ellipsoid Z axis length? (default=1) ",Relay(3)
```

```
5430     END IF
5440     IF Rman THEN
5450       LET Relay(0)=0
5460     ELSE
5470       LET Relay(0)=FRACT(SHIFT(Rptr+Nd2,Nd2)/(Rep+1))!vol frac
5480     END IF
5490     CALL Pixs_by_ellps
5500   CASE =6
5510     REM Relay(0) = volume fractions
5520     REM Relay(1)=seed
5530     REM Relay(2)=correlation select, 0=flat,1=exp,2=inv,3=inv sqr,4=pwr
5540     REM Relay(3)=correlation length or power law
5550     REM Relay(4)=for 1=exp then correl lenght, for 4=pwr then power
5560     IF NOT Rcyc AND Rptr=1 AND Rep>1 THEN
5570       DISP "Corrrelation? none=0 expon=1 inverse=2 inv square=3";
5580       DISP " power law=4 (default=";VAL$(Relay(2));
5590       INPUT ")",Relay(2)
5600       IF Relay(2)=1 THEN
5610         DISP "Correlation length? (in pixel units, default=";
5620         DISP VAL$(Relay(3));
5630         INPUT ")",Relay(3)
5640       END IF
5650       IF Relay(2)=4 THEN
5660         DISP "Power law? (1=inverse 2=inv square, default=";
5670         DISP VAL$(Relay(3));
5680         INPUT ")",Relay(3)
5690       END IF
5700     END IF
5710     IF Rman THEN
5720       Relay(0)=0
5730       Relay(1)=0
5740     ELSE
5750       LET Relay(0)=FRACT(SHIFT(Rptr+Nd2,Nd2)/(Rep+1))!vol frac
5760       LET Relay(1)=Rptr             ! pass to randomize seed
5770     END IF
5780     CALL Pix_by_correlat
5790   CASE =7
5800     REM Relay(0) = volume fractions
5810     REM Relay(1)=seed
5820     REM Relay(2)=number of generations of evolving
5830     IF NOT Rcyc AND Rptr=1 AND Rep>1 THEN
5840       DISP "Maintain constant parent to descendent";
5850       DISP " ratios? Vary=0 Fixed=1 (default=";VAL$(Relay(2));
5860       INPUT ")",Relay(2)
5870       DISP "Number of generations of evolution?";
5880       DISP " (self determine=0, default=";VAL$(Relay(3));
5890       INPUT ")",Relay(3)
5900       DISP "STARTER/CONVERGENCE species? (default=";VAL$(Relay(4));
5910       INPUT ")",Relay(4)
5920       DISP "What is fractional chance of parents descending";
5930       DISP " to other species? (default=";VAL$(FRACT(Relay(5)));
5940       INPUT ")",Relay(5)
5950     END IF
5960     IF Rman THEN
5970       LET Relay(0)=0
```

```
5980        LET Relay(1)=0
5990        LET Relay(2)=0
6000        LET Relay(3)=0
6010        LET Relay(4)=0
6020        LET Relay(5)=0
6030      ELSE
6040        LET Relay(0)=FRACT(SHIFT(Rptr+Nd2,Nd2)/(Rep+1))!vol frac
6050        LET Relay(1)=Rptr            ! pass to randomize seed
6060      END IF
6070      CALL Pixs_by_evolv(Spcs)
6080    END SELECT
6090    LET Infc=0                       !Initialize interface tally
6100    MAT Frk=(0)                      !Initialize volume fractions
6110    FOR Nwrk=1 TO Ndmax-1            !Tally up pixel types
6120      IF Pixl(Nwrk)>0 THEN LET Frk(Pixl(Nwrk))=Frk(Pixl(Nwrk))+1
6130    NEXT Nwrk
6140    MAT Frk=Frk/(Ndmax-1)            !convert to vol fractions
6150    IF Rptr<3 THEN                   !print out pixel array
6160      PRINT "The pixel box array";Xmax;"x";Ymax;"x";Zmax;
6170      PRINT "contains the types:"
6180      FOR Xcnt=1 TO Xmax
6190        PRINT RPT$(" ",Ymax);"X=";VAL$(Xcnt)
6200        FOR Zcnt=Zmax TO 1 STEP -1
6210          FOR Ycnt=1 TO Ymax
6220            LET Nwrk=FNOde_mk(Xcnt,Ycnt,Zcnt,Xmax,Ymax,Zmax)
6230            PRINT USING "DD,#";Pixl(Nwrk)
6240          NEXT Ycnt
6250          PRINT
6260        NEXT Zcnt
6270      NEXT Xcnt
6280    END IF
6290    PRINT " Volume fractions are: "
6300    FOR Xcnt=1 TO Spcs
6310      PRINT "[";VAL$(Xcnt);"]s have ";VAL$(DROUND(Frk(Xcnt),4));
6320      IF Xcnt<Spcs THEN PRINT ",";
6330    NEXT Xcnt
6340    PRINT
6350    PRINT "Species permittivities: ";
6360    FOR Xcnt=1 TO Spcs
6370      PRINT " [";VAL$(Xcnt);"]s=(";VAL$(DROUND(REAL(Dlct(Xcnt)),4));",";
6380      PRINT VAL$(DROUND(IMAG(Dlct(Xcnt)),4));")";
6390    NEXT Xcnt
6400    PRINT
6410    IF Dtl THEN PRINT "Principal node & participating neighbor nodes:"
6420    IF Dtl THEN PRINT " Node [ 0] touches";
6430    LET Hdig=1
6440    FOR Nd2=1 TO Zxy                 !Z=1 level nodes touching ground
6450      IF Dtl THEN PRINT USING """ ["",DD,""]"",#";Nd2
6460      LET Admt0=Dlct(Pixl(Nd2))*Zadm!only path admittance to ground
6470      IF Meth=1 THEN LET Act1(Nd2,Nd2)=Admt0
6480      IF Meth=2 THEN LET Act2(Hdig)=Admt0
6490      LET Hdig=Hdig+Nd2+1            !1,3,6,10,15.. diagonal elements
6500    NEXT Nd2
6510    IF Dtl THEN PRINT
6520    IF Meth=2 THEN
```

```
6530      LET TmvO=TIMEDATE          !clock nodal analysis, Meth=2
6540      DISP " ..wait,";Rptr;"of ";VAL$(SHIFT(Rep,-(Knj>0))+Rman);
6550      DISP ", solve";Ndmax;"nodes >@";TIME$(TmvO)
6560    END IF
6570    LET Lpiv=0                   !pivoting occurances, initialize
6580    LET Hdig=1                   !1,3,6,10,15.. diagonal elements
6590    FOR Nd1=1 TO Ndmax-1         !Range thru all pixel block nodes
6600      IF Dtl THEN PRINT USING """Node ["",DD,""] touches"",#";Nd1
6610      CALL Node_addrs(Nd1,Xadr,Yadr,Zadr,Xmax,Ymax,Zmax)!(X,Y,Z) of node
6620      LET Admt1=Dlct(Pixl(Nd1))  !Admittance in principal node
6630      FOR Axs=1 TO 3             !(1,2,3)<=>(x,y,z) axes
6640        FOR Face=0 TO 1          ! 0=lower 1=upper face wrt choosen axis
6650          LET Nd2=-1             !Intialize as no known node
6660          IF Face THEN           !IF Face(t)=1 (upper face)
6670            SELECT Axs
6680            CASE =1
6690              IF Xadr<Xmax THEN Nd2=Nd1+1 !ordinary neighbor
6700              IF Xadr=Xmax AND Kond=2 THEN Nd2=Nd1-Xmax+1 !wrap periodic X
6710            CASE =2
6720              IF Yadr<Ymax THEN Nd2=Nd1+Xmax
6730              IF Yadr=Ymax AND Kond=2 THEN Nd2=Nd1-Zxy+Xmax !wrap periodic
6740            CASE =3
6750              IF Zadr<Zmax THEN Nd2=Nd1+Zxy
6760              IF Zadr=Zmax THEN Nd2=Ndmax !exciter electrode is neighbor
6770            END SELECT
6780          ELSE                   !else, Face=0 (lower face)
6790            SELECT Axs
6800            CASE =1
6810              IF Xadr>1 THEN Nd2=Nd1-1
6820              IF Xadr=1 AND Kond=2 THEN Nd2=Nd1+Xmax-1 !wrap periodic X
6830            CASE =2
6840              IF Yadr>1 THEN Nd2=Nd1-Xmax
6850              IF Yadr=1 AND Kond=2 THEN Nd2=Nd1+Zxy-Xmax !wrap periodic Y
6860            CASE =3
6870              IF Zadr>1 THEN Nd2=Nd1-Zxy
6880              IF Zadr=1 THEN Nd2=0    !ground electrode is neighbor
6890            END SELECT
6900          END IF                      !end if, Face(t)
6910          IF Dtl THEN PRINT USING """ ["",DD,""]"",#";Nd2
6920          IF Nd2>0 AND Nd1<>Nd2 AND Nd2<Ndmax THEN !a decent neighbor
6930            LET Admt2=Dlct(Pixl(Nd2))!Admittance
6940            LET AdmtO=Admt1+Admt2      ! path admittance
6950            IF Pixl(Nd1)<>Pixl(Nd2) THEN Infc=Infc+1 !tally up interfaces
6960            IF AdmtO<>CMPLX(0,0) THEN
6970              LET AdmtO=Admt1*Admt2/AdmtO
6980              IF Axs=1 THEN AdmtO=AdmtO*Xadm
6990              IF Axs=2 THEN AdmtO=AdmtO*Yadm
7000              IF Axs=3 THEN AdmtO=AdmtO*Zadm
7010              IF Meth=1 THEN
7020                LET Act1(Nd1,Nd1)=Act1(Nd1,Nd1)+AdmtO !self interact
7030                LET Act1(Nd1,Nd2)=Act1(Nd1,Nd2)-AdmtO !neighbor interact
7040              ELSE                   !else Meth=2, hopper/sparseness tech
7050                LET Hrel=Nd2-Nd1 !Nd2's address relv to diag element
7060                IF Nd1<=Hedge THEN !fill hopper
7070                  LET Act2(Hdig)=Act2(Hdig)+AdmtO !Nd1 self interact
```

```
7080                IF Hrel<0 THEN   !ie Nd2<Nd1, neighbor interact
7090                  LET Act2(Hdig+Hrel)=Act2(Hdig+Hrel)-Admt0
7100                END IF
7110              ELSE                 !else w/ hopper/sparseness analysis
7120                LET Act2(Hmem)=Act2(Hmem)+Admt0
7130                IF Hrel<0 AND Hrel+Hedge>0 THEN !good neighbors, go ahead
7140                  LET Act2(Hmem+Hrel)=Act2(Hmem+Hrel)-Admt0
7150                END IF
7160              END IF               ! end if Nd1>Hedge
7170            END IF                 !Meth
7180          END IF                   !On Admittances non-zero
7190        END IF                     !end if, Nd2 is decent neihbor
7200        IF Meth=2 AND Nd2=Ndmax THEN !if path to exciter electrode
7210          LET Act2(Hmem)=Act2(Hmem)+Dlct(Pixl(Nd1))*Zadm
7220        END IF
7230      NEXT Face
7240    NEXT Axs
7250    IF Dtl THEN PRINT
7260    IF Dtl AND Meth=2 THEN
7270      PRINT USING """ Feed ["",DD,""]"",#";Nd1
7280      IF Nd1<=Hedge THEN Hlmt=Hdig-Nd1+1
7290      IF Nd1>Hedge THEN Hlmt=Hmem-Hedge+1
7300      FOR Hclm=Hlmt TO Hdig
7310        PRINT USING """("",4A,#";VAL$(REAL(Act2(Hclm)))
7320        PRINT USING ""","",4A,"")"",#";VAL$(IMAG(Act2(Hclm)))
7330        IF Hclm-Hlmt MOD 6=5 THEN PRINT !start a new line
7340      NEXT Hclm
7350      IF Hclm-Hlmt MOD 6<>0 THEN PRINT
7360    END IF                         !end if, Dtl
7370    IF Meth=2 AND Nd1>=Hedge THEN !start hopper reduction
7380      LET Lpiv=Lpiv+1              !# pivoting occurances
7390      MAT Hpiv=(CMPLX(0,0))
7400      LET Hpiv(1)=CMPLX(1,0)
7410      IF Act2(1)<>CMPLX(0,0) THEN !initialize pivoting vector
7420        LET Hnrm=1/Act2(1)         !normalizing factor
7430        LET Hclm=2                 !relative address counter
7440        IF Dtl THEN PRINT " Piv {";VAL$(Lpiv);"} (1,0)";
7450        FOR Hrow=2 TO Hedge        !set pivot vector
7460          LET Hpiv(Hrow)=Act2(Hclm)*Hnrm
7470          IF Back THEN Tca2(Hrow-1,Lpiv)=Hpiv(Hrow)
7480          IF Dtl THEN              !print out pivots
7490            PRINT USING """("",4A,#";VAL$(REAL(Hpiv(Hrow)))
7500            PRINT USING ""","",4A,"")"",#";VAL$(IMAG(Hpiv(Hrow)))
7510          END IF
7520          LET Hclm=Hclm+Hrow       !2,4.7,11,16.. row start
7530        NEXT Hrow
7540        IF Dtl THEN PRINT
7550      END IF                       !end if, pivot setting
7560      LET Hcnt=1                   !end of row counter
7570      FOR Hrow=2 TO Hedge          !heart of pivoting
7580        IF Hpiv(Hrow)<>CMPLX(0,0) THEN !go ahead
7590          FOR Hclm=2 TO Hrow       !adj @ row with pivots
7600            LET Nmr=Hcnt+Hclm
7610            LET Act2(Nmr)=Act2(Nmr)-Act2(Hcnt+1)*Hpiv(Hclm)
7620          NEXT Hclm
```

```
7630          END IF
7640          LET Hcnt=Hcnt+Hrow        !1,3,6,10.. gives row ends
7650        NEXT Hrow
7660        IF Dtl THEN
7670          LET Hcnt=0
7680          FOR Nmr=1 TO Hedge        !dump Act2 array
7690            PRINT USING """  Hop act<"",DD,"">"",#";Nmr
7700            FOR Hclm=1 TO Nmr
7710              PRINT USING """("",4A,#";VAL$(REAL(Act2(Hclm+Hcnt)))
7720              PRINT USING ""","",4A,"")"",#";VAL$(IMAG(Act2(Hclm+Hcnt)))
7730              IF Hclm MOD 6=0 THEN PRINT  !start a new line
7740            NEXT Hclm
7750            IF Hclm MOD 6<>1 THEN PRINT
7760            LET Hcnt=Hcnt+Nmr       !0,1,3,6,10,15.. diags
7770          NEXT Nmr
7780        END IF                      !end if Dtl, Act array dump
7790        LET Hcnt=0                  !ref address to end of previous row
7800        FOR Hrow=1 TO Hedge-1       !shakedown for hopper
7810          FOR Hclm=1 TO Hrow
7820            LET Nmr=Hcnt+Hclm
7830            LET Act2(Nmr)=Act2(Nmr+1+Hrow)
7840          NEXT Hclm
7850          LET Hcnt=Hcnt+Hrow        !0,1,3,6,10,15.. row ends
7860        NEXT Hrow                   !ready to feed hooper
7870        FOR Hclm=Hmem+1-Hedge TO Hmem
7880          LET Act2(Hclm)=CMPLX(0,0) !zero to feed new elements
7890        NEXT Hclm
7900      END IF                        !end if for Ndl>Hedge
7910      IF Ndl<Hedge THEN Hdig=Hdig+Ndl+1 !1,3,6,10.. row end
7920    NEXT Ndl
7930    LET Ndl=Ndmax                   !Exciter node
7940    IF Dtl THEN PRINT USING """Node ["",DD,""] touches"",#";Ndl
7950    IF Back THEN MAT Endex=(CMPLX(0,0))  !initialize exciter interact
7960    FOR Nd2=Ndmax-Zxy TO Ndmax-1    !Z=Zmax level nodes to exciter
7970      IF Dtl THEN PRINT USING """ ["",DD,""]"",#";Nd2
7980      LET Admt0=Dlct(Pixl(Nd2))*Zadm!path admittance to exciter
7990      LET Hrel=Nd2-Ndl              !relative hopper address
8000      IF Meth=1 THEN
8010        LET Act1(Ndl,Ndl)=Act1(Ndl,Ndl)+Admt0
8020        LET Act1(Nd2,Nd2)=Act1(Nd2,Nd2)+Admt0
8030        LET Act1(Ndl,Nd2)=Act1(Ndl,Nd2)-Admt0
8040        LET Act1(Nd2,Ndl)=Act1(Nd2,Ndl)-Admt0
8050      ELSE
8060        LET Act2(Hmem)=Act2(Hmem)+Admt0
8070        IF Hrel+Hedge>0 THEN           !good neighbor to interact
8080          LET Act2(Hmem+Hrel)=Act2(Hmem+Hrel)-Admt0
8090        END IF
8100      END IF                           !end if Meth
8110      IF Back THEN                     !save exciter interactions
8120        LET Endex(Zxy+1)=Endex(Zxy+1)+Admt0 !self exciter
8130        LET Endex(Zxy+1+Hrel)=Endex(Zxy+1+Hrel)-Admt0 !neighbor to exciter
8140      END IF
8150    NEXT Nd2
8160    IF Dtl THEN PRINT
8170    IF Dtl AND Meth=2 THEN
```

```
8180       PRINT USING """ Feed ["",DD,""]"",#";Ndl
8190       LET Hlmt=Hmem-Hedge+1
8200       FOR Hclm=Hlmt TO Hdig
8210         PRINT USING """("",4A,#";VAL$(REAL(Act2(Hclm)))
8220         PRINT USING ""","",4A,"")"",#";VAL$(IMAG(Act2(Hclm)))
8230         IF Hclm-Hlmt MOD 6=5 THEN PRINT   !start a new line
8240       NEXT Hclm
8250       IF Hclm-Hlmt MOD 6<>0 THEN PRINT
8260     END IF                              ! end if Dtl
8270     IF Meth=1 THEN
8280       LET Tmv0=TIMEDATE                 !Clock nodal analysis, Meth=1
8290       DISP " ..wait,";Rptr;"of ";VAL$(SHIFT(Rep,-(Knj>0))+Rman);
8300       DISP ", solve";Ndmax;"nodes from ";TIME$(Tmv0)
8310       MAT Tcal=INV(Act1)                !solving nodal analysis matrix
8320     ELSE
8330       FOR Hlmt=Hedge TO 2 STEP -1  !play out hopper to funnel it down
8340         LET Lpiv=Lpiv+1                 !# of pivoting occurances
8350         MAT Hpiv=(CMPLX(0,0))
8360         LET Hpiv(1)=CMPLX(1,0)
8370         IF Act2(1)<>CMPLX(0,0) THEN !initialize pivoting vector
8380           LET Hnrm=1/Act2(1)        !normalizing factor
8390           IF Dtl THEN PRINT " Piv {";VAL$(Lpiv);"} (1,0)";
8400           LET Hclm=2                    !relative address counter
8410           FOR Hrow=2 TO Hlmt        !set pivot vector
8420             LET Hpiv(Hrow)=Act2(Hclm)*Hnrm
8430             IF Back THEN Tca2(Hrow-1,Lpiv)=Hpiv(Hrow)
8440             IF Dtl THEN                 !print out pivots
8450               PRINT USING """("",4A,#";VAL$(REAL(Hpiv(Hrow)))
8460               PRINT USING ""","",4A,"")"",#";VAL$(IMAG(Hpiv(Hrow)))
8470             END IF
8480             LET Hclm=Hclm+Hrow      !2,4,7,11,16.. row starts
8490           NEXT Hrow
8500           IF Dtl THEN PRINT
8510         END IF                          !end if, pivot setting
8520         LET Hcnt=1                      !end of row counter
8530         FOR Hrow=2 TO Hlmt          !heart of pivoting
8540           IF Hpiv(Hrow)<>CMPLX(0,0) THEN !go ahead
8550             FOR Hclm=2 TO Hrow      !adj @ row with pivots
8560               LET Nmr=Hcnt+Hclm
8570               LET Act2(Nmr)=Act2(Nmr)-Act2(Hcnt+1)*Hpiv(Hclm)
8580             NEXT Hclm
8590           END IF
8600           LET Hcnt=Hcnt+Hrow        !1,3,6,10,15.. row ends
8610         NEXT Hrow
8620         IF Dtl THEN
8630           LET Hcnt=0
8640           FOR Nmr=1 TO Hlmt         !dump Act2 array
8650             PRINT USING """  Hop act<"",DD,"">"",#";Nmr
8660             FOR Hclm=1 TO Nmr
8670               PRINT USING """("",4A,#";VAL$(REAL(Act2(Hclm+Hcnt)))
8680               PRINT USING ""","",4A,"")"",#";VAL$(IMAG(Act2(Hclm+Hcnt)))
8690               IF Hclm MOD 6=0 THEN PRINT  !start a new line
8700             NEXT Hclm
8710             IF Hclm MOD 6<>1 THEN PRINT
8720             LET Hcnt=Hcnt+Nmr       !0,1,3,6,10,15.. diags
```

```
8730          NEXT Nmr
8740        END IF                     !end if Dtl, Act array dump
8750        LET Hcnt=0                 !ref address to end of previous row
8760        FOR Hrow=1 TO Hlmt-1       !shakedown for hopper
8770          FOR Hclm=1 TO Hrow
8780            LET Nmr=Hcnt+Hclm
8790            LET Act2(Nmr)=Act2(Nmr+1+Hrow)
8800          NEXT Hclm
8810          LET Hcnt=Hcnt+Hrow       !0,1,3,6,10,15.. row ends
8820        NEXT Hrow
8830        FOR Hclm=Hcnt+1 TO Hcnt+Hlmt
8840          LET Act2(Hclm)=CMPLX(0,0)!zero last Hopper row
8850        NEXT Hclm
8860      NEXT Hlmt
8870    END IF                         !end if Meth choices
8880    LET Tmv1=TIMEDATE
8890    DISP ">";TIME$(Tmv1-Tmv0);"<";
8900    IF Dtl THEN
8910      PRINT "Nodal analysis solution required elapsed time of ";
8920      PRINT TIME$(Tmv1-Tmv0);"."
8930    END IF
8940    IF Dtl AND Meth=1 THEN         !Printout of nodal analysis array
8950      PRINT "Contents of classic nodal analysis array";Ndmax;"x ";
8960      PRINT VAL$(Ndmax);", starting at node [1,1].."
8970      FOR Nd1=1 TO Ndmax
8980        LET Admt2=CMPLX(0,0)       !initialize nodal row accumulation
8990        PRINT USING """["",DD,"",*]"",#";Nd1
9000        FOR Nd2=1 TO Ndmax
9010          LET Admt2=Admt2+Act1(Nd1,Nd2)
9020          PRINT USING """("",4A,#";VAL$(REAL(Act1(Nd1,Nd2)))
9030          PRINT USING ""","",4A,"")"",#";VAL$(IMAG(Act1(Nd1,Nd2)))
9040          IF Nd2 MOD 6=0 THEN PRINT !start a new line
9050        NEXT Nd2
9060        PRINT USING """@("",5A,#";VAL$(REAL(Admt2))
9070        PRINT USING ""","",5A,"")""";VAL$(IMAG(Admt2))
9080      NEXT Nd1
9090    END IF                                !end if Dtl & Meth=1
9100    IF Dtl AND Meth=2 THEN
9110      PRINT USING """  FinalHopAct<"",DD,"">"",#";Nmr
9120      PRINT USING """("",5A,#";VAL$(REAL(Act2(1)))
9130      PRINT USING ""","",5A,"")""";VAL$(IMAG(Act2(1)))
9140    END IF                                ! end if Dtl
9150    LET Hnrm=CMPLX(1,0)                   ! anticipated normed current
9160    IF Meth=1 THEN
9170      MAT Ptn=Tcal(*,Ndmax)               !potentials, exciter at (1,~)
9180      IF Back THEN                        !check on exciter
9190        LET Hnrm=CMPLX(0,0)               !node current
9200        FOR Nd1=0 TO Zxy
9210          LET Hnrm=Hnrm+Ptn(Ndmax-Nd1)*Endex(Zxy+1-Nd1)
9220        NEXT Nd1
9230      END IF
9240      LET Resp=Tcal(Ndmax,Ndmax)          !Resultant permittivity
9250      IF Resp<>CMPLX(0,0) THEN LET Resp=2*Norm/Resp
9260    ELSE                                  !else do Meth=2
9270      IF Back THEN
```

```
9280        MAT Ptn=(CMPLX(0,0))
9290        IF Act2(1)<>CMPLX(0,0) THEN LET Ptn(Ndmax)=1/Act2(1)
9300        FOR Nd1=Lpiv TO 1 STEP -1          !backtrack node potentials
9310          LET Hlmt=MIN(Hedge-1,Ndmax-Nd1)
9320          FOR Hclm=1 TO Hlmt
9330            LET Nmr=Nd1+Hclm               !Tca2 node number
9340            LET Ptn(Nd1)=Ptn(Nd1)-Tca2(Hclm,Nd1)*Ptn(Nmr)
9350          NEXT Hclm
9360        NEXT Nd1
9370        LET Hnrm=CMPLX(0,0)                !node current
9380        FOR Nd1=0 TO Zxy
9390          LET Hnrm=Hnrm+Ptn(Ndmax-Nd1)*Endex(Hedge-Nd1)
9400        NEXT Nd1
9410      END IF                               !end if, back
9420      LET Resp=2*Norm*Act2(1)              !Resultant permittivity
9430    END IF                                 !end if Meth
9440    IF Back THEN Resp=Resp*Hnrm            !backsubstition correction
9450    IF Back OR Meth=1 THEN                 !presumably arrays exist
9460      MAT Fluz=(CMPLX(0,0))                !for displacement flux lines
9470      FOR Nd1=1 TO Ndmax-1
9480        LET Admt1=Dlct(Pixl(Nd1))
9490        LET Nd2=Nd1-Zxy                    !lower node neighbor
9500        IF Nd2<1 THEN
9510          LET Fluz(Nd1)=.5*Admt1*Ptn(Nd1) !admittance to ground
9520        ELSE
9530          LET Admt2=Dlct(Pixl(Nd2))        !neighbor admittance
9540          LET Admt0=Admt1+Admt2
9550          IF Admt0<>CMPLX(0,0) THEN
9560            LET Admt0=Admt1*Admt2/Admt0
9570            LET Admt0=(Ptn(Nd1)-Ptn(Nd2))*Admt0
9580            LET Fluz(Nd1)=.5*Admt0
9590          END IF
9600        END IF                             !end if, Nd2<1
9610        LET Nd2=Nd1+Zxy                    !upper node neighbor
9620        IF Nd2>=Ndmax THEN                 !admittance to exciter
9630          LET Fluz(Nd1)=Fluz(Nd1)+.5*Admt1*(Ptn(Ndmax)-Ptn(Nd1))
9640        ELSE
9650          LET Admt2=Dlct(Pixl(Nd2))         !neighbor admittance
9660          LET Admt0=Admt1+Admt2
9670          IF Admt0<>CMPLX(0,0) THEN
9680            LET Admt0=Admt1*Admt2/Admt0
9690            LET Admt0=(Ptn(Nd2)-Ptn(Nd1))*Admt0
9700            LET Fluz(Nd1)=Fluz(Nd1)+.5*Admt0
9710          END IF
9720        END IF                             !end if, Nd2>=Ndmax
9730      NEXT Nd1
9740    END IF                                 !end if, Back or Meth=1
9750    IF Rman AND (Meth=1 OR Back) THEN
9760      IF REAL(Ptn(Ndmax))<>0 THEN MAT Ptn=Ptn/(REAL(Ptn(Ndmax)))
9770      PRINT "Potentials: at exciter node ";
9780      PRINT USING """(""",5A,#";VAL$(REAL(Ptn(Ndmax)))
9790      PRINT USING """,""",5A,"")""";VAL$(IMAG(Ptn(Ndmax)))
9800      FOR Xcnt=1 TO Xmax
9810        PRINT " Cross section, X=";VAL$(Xcnt)
9820        FOR Zcnt=Zmax TO 1 STEP -1
```

```
9830          FOR Ycnt=1 TO Ymax
9840            LET Nwrk=FNOde_mk(Xcnt,Ycnt,Zcnt,Xmax,Ymax,Zmax)
9850            PRINT USING """(""",5A,#";VAL$(REAL(Ptn(Nwrk)))
9860            PRINT USING """,""",5A,"")"",#";VAL$(IMAG(Ptn(Nwrk)))
9870            IF Ycnt MOD 5=0 THEN PRINT
9880          NEXT Ycnt
9890          IF Ycnt MOD 5<>1 THEN PRINT
9900        NEXT Zcnt
9910      NEXT Xcnt
9920      PRINT "Displacement fluxes:  exciter node normalized to (1,0)"
9930      LET Admt2=CMPLX(0,0)
9940      FOR Xcnt=1 TO Xmax
9950        PRINT " Cross section, X=";VAL$(Xcnt)
9960        FOR Zcnt=Zmax TO 1 STEP -1
9970          FOR Ycnt=1 TO Ymax
9980            LET Nwrk=FNOde_mk(Xcnt,Ycnt,Zcnt,Xmax,Ymax,Zmax)
9990          ! PRINT USING """(""",5A,#";VAL$(REAL(Fluz(Nwrk)))
10000         ! PRINT USING """,""",5A,"")"",#";VAL$(IMAG(Fluz(Nwrk)))
10010           LET Nmr=INT(.5+4*Zxy*REAL(Fluz(Nwrk)))
10020           LET Nd2=7-SHIFT(Nmr+1,1)
10030           IF Nd2>0 THEN Nmr=Nmr+Nd2  !roughly centered
10040           FOR Nd1=1 TO 13            !print real flux
10050             IF Nd1<Nd2 THEN PRINT " ";
10060             IF Nd1>=Nd2 AND Nd1<Nmr THEN PRINT """";
10070             IF Nd1>Nmr THEN PRINT " ";
10080           NEXT Nd1
10090           IF Ycnt MOD 5=0 THEN PRINT
10100           LET Admt2=Admt2+Fluz(Nwrk)
10110         NEXT Ycnt
10120         IF Ycnt MOD 5<>1 THEN PRINT
10130       NEXT Zcnt
10140     NEXT Xcnt
10150     LET Admt2=Admt2/Zmax
10160     PRINT "Average displacement flux through a plane";
10170     PRINT " perpendicular to the electrodes is "
10180     PRINT "  (";VAL$(DROUND(REAL(Admt2),4));
10190     PRINT ",";VAL$(DROUND(IMAG(Admt2),4));")"
10200   END IF                          ! end if for Rman output
10210   PRINT " Resultant permittivity is (";REAL(Resp);",";IMAG(Resp);") & it's"
10220   LET Alph=FNWnr(Dlct(*),Frk(*),Resp,Iterr,Spcs)
10230   IF IMAG(Alph)<>Relay(7) THEN
10240     IF REAL(Alph)<>Relay(6) THEN
10250       PRINT " fail? to pass both cmplx parts of exp av";
10260     ELSE
10270       PRINT " fail? to pass imag part of the exp ave";
10280     END IF
10290     PRINT " factr via FNWnr, now using Relay(6&7)"
10300     LET Alph=CMPLX(Relay(6),Relay(7))
10310   END IF
10320   PRINT " exponential averaging (also Wiener or series<->parallel)";
10330   PRINT " factor is"
10340   PRINT "  (";REAL(Alph);",";IMAG(Alph);")."
10350   IF Ndmax>1 THEN PRINT " Interfaces/pixel=";VAL$(DROUND(Infc/(Ndmax-1),4));
10360   IF Back THEN PRINT " & backsubst correction=";VAL$(ABS(Hnrm-CMPLX(1,0)))
10370   REM Output each cycle to internal storage array
```

```
10380   IF Knj=2 THEN                    ! averaged cases of conjugation pairs
10390     LET Nwrk=SHIFT(Rptr+1,1)       ! effective 1,1,2,2,3,3..
10400     LET Graf(Nwrk,1)=(Nwrk+Iterr)*.5+Graf(Nwrk,1)
10410     LET Graf(Nwrk,2)=Frk(1)*.5+Graf(Nwrk,2)
10420     LET Graf(Nwrk,3)=REAL(Alph)*.5+Graf(Nwrk,3)
10430     LET Graf(Nwrk,4)=IMAG(Alph)*.5+Graf(Nwrk,4)
10440     LET Graf(Nwrk,5)=REAL(Resp)*.5+Graf(Nwrk,5)
10450     LET Graf(Nwrk,6)=IMAG(Resp)*.5+Graf(Nwrk,6)
10460     IF Ndmax>1 THEN LET Graf(Nwrk,7)=Infc/(Ndmax-1)
10470     IF Ptrn=5 THEN                  !other information programmed
10480       LET Graf(Nwrk,8)=Relay(4)*.5+Graf(Nwrk,8) !ellipse roles
10490     ELSE
10500       LET Graf(Nwrk,8)=Relay(2)*.5+Graf(Nwrk,8)
10510     END IF
10520   ELSE                             ! consequetive cases, no averaging
10530     LET Graf(Rptr,1)=Rptr+Iterr
10540     LET Graf(Rptr,2)=Frk(1)
10550     LET Graf(Rptr,3)=REAL(Alph)
10560     LET Graf(Rptr,4)=IMAG(Alph)
10570     LET Graf(Rptr,5)=REAL(Resp)
10580     LET Graf(Rptr,6)=IMAG(Resp)
10590     IF Ndmax>1 THEN LET Graf(Rptr,7)=Sgj*Infc/(Ndmax-1)
10600     IF Ptrn=5 THEN                 ! other programmed information
10610       LET Graf(Rptr,8)=Relay(4)    ! ellipse role
10620     ELSE
10630      !LET Graf(Rptr,8)=Relay(2)    ! shuffle choice
10640       LET Graf(Rptr,8)=Tmv1-Tmv0   ! matrix inversion time
10650     END IF
10660   END IF                           ! end if, Knj=2
10670   IF Rep>0 THEN SOUND 1,100,10,.04 ! low audible tone per Rptr cycle
10680   LET Tmup=Tmcyc+(TIMEDATE-Tmcyc)*(SHIFT(Rep,-(Knj>0))+Rman)/Rptr+4
10690   IF Knj>0 THEN                    !conjugate on altn Rptr after data done
10700     MAT Dlct=CONJG(Dlct)
10710     LET Sgj=-Sgj
10720   END IF
10730   DISP " finis ~";TIME$(Tmup);" ";
10740   DISP USING "AAAAAA,#";DATE$(Tmup)
10750 NEXT Rptr
10760 IF Sgj=-1 THEN                     !hopefully unCONJGates
10770   MAT Dlct=CONJG(Dlct)
10780   LET Sgj=1
10790 END IF
10800 !   =   =   =   =   S U M M A R Y   O U T P U T   =   =   =   =   =
10810 DISP
10820 LET Bhdr$="PIXonNODE "
10830 LET Bhdr$=Bhdr$&"SIZ=["&VAL$(Xmax)&"x"&VAL$(Ymax)&"x"&VAL$(Zmax)&"] "
10840 LET Bhdr$=Bhdr$&"SCL="&VAL$(DROUND(Xscl,3))&","
10850 LET Bhdr$=Bhdr$&VAL$(DROUND(Yscl,3))&","&VAL$(DROUND(Zscl,3))&" "
10860 IF Kond=1 THEN Bhdr$=Bhdr$&"InslBC "
10870 IF Kond=2 THEN Bhdr$=Bhdr$&"PrdcBC "
10880 IF Meth=1 THEN Bhdr$=Bhdr$&"Trad "
10890 IF Meth=2 THEN Bhdr$=Bhdr$&"Spar "
10900 IF Ptrn=0 THEN Bhdr$=Bhdr$&"from prev "
10910 IF Ptrn=1 THEN Bhdr$=Bhdr$&"from """&Fln$&""" "
10920 IF Ptrn=2 THEN Bhdr$=Bhdr$&"from USER "
```

```
10930 IF Ptrn=3 THEN Bhdr$=Bhdr$&"RANDOM "
10940 IF Ptrn=4 THEN Bhdr$=Bhdr$&"FRACTAL "
10950 IF Ptrn=5 THEN Bhdr$=Bhdr$&"ELLIPS "
10960 IF Knj=0 THEN Bhdr$=Bhdr$&"NO * "
10970 IF Knj=1 THEN Bhdr$=Bhdr$&"ALT* "
10980 IF Knj=2 THEN Bhdr$=Bhdr$&"AVG* "
10990 IF LEN(Bhdr$)<80 THEN Bhdr$[1+LEN(Bhdr$)]=RPT$(" ",80-LEN(Bhdr$))
11000 LET Trs$=TIME$(TIMEDATE)
11010 LET Ws$=DATE$(TIMEDATE)&" "&Trs$[1;5]
11020 LET Bhdr$[81-LEN(Ws$)]=Ws$          ! Tack time & date to end
11030 IF Rep>1 THEN
11040   PRINT "Summary of repeat cycles: (may be reprogrammed)"
11050   PRINT " TRY.err, VOL FRACTs, REAL, IMAG ALPHA,";
11060   PRINT " REAL, IMAG PERMITTIVITY, & more"
11070   FOR Rptr=1 TO SHIFT(Rep,-(Knj=1))
11080     PRINT "#";VAL$(Rptr);".)";
11090     FOR Xcnt=1 TO Kwd
11100       PRINT USING "X,10A,#";VAL$(DROUND(Graf(Rptr,Xcnt),4))
11110     NEXT Xcnt
11120     PRINT
11130   NEXT Rptr
11140   IF Svr THEN
11150     LET Nmr=SIZE(Graf,1)*SIZE(Graf,2) !memory elements (@8 bytes)
11160     IF Ovr=3 THEN CREATE Fln$,Nmr*8.0+256.0 !for HFS
11170    !IF Ovr=3 THEN CREATE Fln$,1      !for LIF or DOS
11180     LET Nmr=SHIFT(Rep,-(Knj=1))      !length of array Graf
11190     REDIM Graf(1:Nmr,1:Kwd)          !REDIM necessary to size COM output
11200     ASSIGN @Savgraf TO Fln$;FORMAT OFF
11210     IF Ovr=1 OR Ovr=3 THEN
11220       OUTPUT @Savgraf;Ahdr$,Bhdr$,Nmr,Kwd,Graf(*)
11230       OUTPUT @Savgraf;Dlct(*),Xmax,Ymax,Zmax,Xscl,Yscl,Zscl,END
11240     END IF
11250     IF Ovr=2 THEN                    ! append only
11260       ON END @Savgraf GOTO 11300
11270       REM read until end
11280       ENTER @Savgraf;Nwrk            ! read by integers (ie 2 bytes)
11290       GOTO 11270
11300       REM at file end now append
11310       OFF END @Savgraf
11320       OUTPUT @Savgraf;Ahdr$,Bhdr$,Nmr,Kwd,Graf(*)
11330       OUTPUT @Savgraf;Dlct(*),Xmax,Ymax,Zmax,Xscl,Yscl,Zscl,END
11340     END IF                           ! end if to append
11350     ASSIGN @Savgraf TO *
11360   END IF
11370 END IF
11380 LET Chdr$=Ahdr$                     !save titling
11390 LET Dhdr$=Bhdr$
11400 LET Mem1=INT(.5+VAL(SYSTEM$("AVAILABLE MEMORY")))
11410 PRINT "New memory used is";PROUND((Mem0-Mem1)/16,2);"& memory";
11420 PRINT " remaining is";PROUND(Mem1/16,2);"in complex units."
11430 LET Tm1=TIMEDATE
11440 PRINT "TOTAL ELAPSED TIME: ";TIME$(Tm1-Tm0)
11450 PRINT "Date: ";DATE$(Tm1);RPT$(" _ ",7);"FINIS";
11460 PRINT RPT$(" _ ",7);"Time: ";TIME$(Tm1)
11470 SOUND 1,132,14,.2                   ! last call, tell user done
```

```
11480 WAIT .25
11490 SOUND 1,110,15,.5
11500 DISP "Last nodal analysis solution cycle required elapsed time of ";
11510 DISP TIME$(Tmv1-Tmv0);"."
11520 END
11530 !   [    ]    [    ]    [   FNOde_mk   ]    [    ]    [    ]    [    ]
11540 DEF FNOde_mk(INTEGER Xmk,Ymk,Zmk,Xbig,Ybig,Zbig)
11550 ! This function returns the node number for the box Pixel grid
11560 ! Xmk,Ymk,Zmk = the address coordinates of the pixel
11570 ! Xbig,Ybig,Zbig = the Pixel extent in each direction
11580   INTEGER Ndmk! Node number
11590   IF Zmk>0 AND Zmk<=Zbig THEN Ndmk=Xmk+((Ymk-1)+(Zmk-1)*Ybig)*Xbig
11600   IF Zmk<0 THEN Ndmk=0                 !Ground electrode
11610   IF Zmk>Zbig THEN Ndmk=Xbig*Ybig*Zbig+1 !Exciter electrode
11620   IF Xmk<1 OR Xmk>Xbig OR Ymk<1 OR Ymk>Ybig THEN Ndmk=-1
11630   RETURN Ndmk
11640 FNEND
11650 !   [    ]    [    ]    [    SUB Nodeaddrs    [    ]    [    ]    [    ]
11660 SUB Node_addrs(INTEGER Ndsrc,Xxx,Yyy,Zzz,Xlmt,Ylmt,Zlmt)
11670 ! This subroutine returns the (X,Y,Z) addresses for a node number
11680 ! Ndsrc = the node number IN variable
11690 ! Xxx,Yyy,Zzz = the Pixel addresses in each direction OUT variable
11700 ! Xlmt,Ylmt,Zlmt = the Pixel extent in each direction IN variable
11710 ! Sqrxy = the size of a XY plane
11720 ! Ndnd = maximum node
11730 ! Rrr = remaiders, work integer
11740   INTEGER Rrr,Ndnd,Sqrss
11750   LET Sqrxy=Xlmt*Ylmt
11760   LET Ndnd=Sqrxy*Zlmt+1
11770   IF Ndsrc>0 AND Ndsrc<Ndnd THEN
11780     LET Zzz=1+(Ndsrc-1) DIV Sqrxy
11790     LET Rrr=(Ndsrc-1) MOD Sqrxy
11800     LET Yyy=1+(Rrr DIV Xlmt)
11810     LET Xxx=1+(Rrr MOD Xlmt)
11820   ELSE
11830     LET Xxx=SHIFT(Xlmt,1)
11840     LET Yyy=SHIFT(Ylmt,1)
11850     IF Ndsrc>=Ndnd THEN Zzz=Zlmt+1
11860     IF Ndsrc<1 THEN Zzz=0
11870   END IF
11880 SUBEND
11890 !   [    ]    [    ]    function FNWnr    [    ]    [    ]    [    ]
11900 DEF FNWnr(COMPLEX Diel(*),REAL Frpx(*),COMPLEX Din,REAL Alferr,INTEGER Nth)
11910   REM  Object of this function subprogram is to
11920   REM  find the exponential averaging factor (or
11930   REM  percolation related factor) "alf"
11940   REM  from a given set of complex number
11950   REM  dielectric values & fractional volume
11960   REM  weights and effective or resultant
11970   REM  complex dielectric value of composite
11980   REM           written by S. Wallin, 4/91.
11990   REM  The Wiener or exponential averaging factor
12000   REM  is defined as follows:
12010   REM  Diel0(resultant)^Alf = sum {Frpx(k)*Diel(k)^Alf}
12020   REM   where Diel0(resultant) = response of composite
```

```
12030   REM         Alf = exponential ave or Wiener or percolation factor
12040   REM         Frpx(k) = fractional volumes of species k
12050   REM         Diel(k) = (dielectric) response of species k
12060   COM /Pass/Relay(0:7)
12070   INTEGER I,J,K,K1,K2,Kdo,Ns,Lsn,Lst,New
12080   COMPLEX Diel0,Dlog0,Alf,Alf0,C0,C1,C2,C3,Clg,Clg2
12090   LET Ns=Nth
12100   IF Ns<=0 THEN STOP
12110   ALLOCATE COMPLEX Dlog(Ns)
12120   LET Avg=0
12130   LET Diel0=Din
12140   FOR I=1 TO Ns
12150     IF Diel(I)<>CMPLX(0,0) THEN Avg=Avg+Frpx(I)
12160   NEXT I
12170   REM Normalize ACTIVE volume to total 1
12180   FOR I=1 TO Ns
12190     !IF Avg<>0 THEN LET Frpx(I)=Frpx(I)/Avg
12200     IF Diel(I)=CMPLX(0,0) THEN Frpx(I)=0
12210   NEXT I
12220 !PRINT " Species data: (trial#, complex ";CHR$(238);" pair, adj vol wt)"
12230 !FOR I=1 TO Ns
12240  !PRINT " [#";I;"] (";REAL(Diel(I));",";IMAG(Diel(I));")",DROUND(Frpx(I),4)
12250 !NEXT I
12260 !PRINT " [ eff] (";REAL(Diel0);",";IMAG(Diel0);")",1
12270   REM Determination of slope direction by log wt
12280   LET Dlog0=CMPLX(0,0)
12290   IF Diel0<>CMPLX(0,0) THEN Dlog0=LOG(Diel0)
12300   LET Clg=CMPLX(0,0)! Clg=Logarthimic mean
12310   LET Clg2=CMPLX(0,0)
12320   FOR I=1 TO Ns
12330     LET Dlog(I)=CMPLX(0,0)
12340     IF Diel(I)<>CMPLX(0,0) THEN Dlog(I)=LOG(Diel(I))-Dlog0
12350     LET Clg=Clg+Frpx(I)*Dlog(I)
12360     LET Clg2=Clg2+Frpx(I)*Dlog(I)*Dlog(I)
12370   NEXT I
12380   LET Lsn=-SGN(REAL(Clg))
12390 !PRINT " The logarithmic slope is ";DROUND(REAL(Clg),4);
12400 !PRINT DROUND(IMAG(Clg),4);"indicates ";CHR$(224);" is ";
12410 !IF Lsn=1 THEN !PRINT "positive."
12420 !IF Lsn=0 THEN !PRINT "at zero."
12430 !IF Lsn=-1 THEN !PRINT "negative."
12440   REM Extrema values
12450   LET Lst=0
12460   LET Zst=0
12470   FOR I=1 TO Ns
12480     IF Frpx(I)<>0 THEN
12490       LET Tmp=Lsn*ABS(Diel(I))
12500       IF Tmp>Zst OR Zst=0 THEN
12510         LET Zst=Tmp
12520         LET Lst=I
12530       END IF
12540     END IF
12550   NEXT I
12560   IF Lsn=0 THEN Lst=0
12570   LET Alf=CMPLX(0,0)
```

```
12580   IF Lst>0 AND Lst<=Ns+1 THEN
12590     IF Dlog(Lst)<>CMPLX(0,0) THEN Alf=-LOG(Frpx(Lst))/Dlog(Lst)
12600   END IF
12610   LET C0=CMPLX(Lsn,0)
12620   IF Clg2<>CMPLX(0,0) THEN C0=-2*Clg/Clg2!A 2nd guess
12630   LET Wt=ABS(C0)
12640   LET Wt=1/(1+Wt*Wt)!Relative weights for ave the 2 guesses
12650   LET Alf=Alf+Wt*(C0-Alf)!Combined 1st guess
12660  !PRINT "   Guess 1   ";CHR$(224);" = ";
12670  !PRINT DROUND(REAL(Alf),4);DROUND(IMAG(Alf),4)
12680   LET Alf0=CMPLX(0,0)
12690   LET Alferr=1
12700   LET J=2
12710   LET New=0
12720   WHILE Alf<>Alf0 AND J<32 AND Alferr>1.0E-13
12730     IF New=1 THEN
12740       LET Alf0=Alf!Keep track of last iteration
12750       New=0
12760     END IF
12770     LET C1=CMPLX(0,0)
12780     LET C2=CMPLX(0,0)
12790     LET C3=CMPLX(0,0)
12800     LET K=0!Keep count of non-zero terms
12810     FOR I=1 TO Ns
12820       LET C0=Alf*Dlog(I)
12830       IF ABS(REAL(C0))>700 THEN !Failure possible
12840         LET Alf=-2*Clg/Clg2
12850         LET Alf0=CMPLX(0,0)
12860         LET C0=CMPLX(0,0)
12870         LET C1=CMPLX(0,0)
12880         LET Alferr=0!Set to exit
12890       END IF
12900       IF C0<>CMPLX(0,0) AND Diel(I)<>0 THEN
12910         LET C0=Frpx(I)*EXP(C0)
12920         LET C1=C1+C0
12930         LET C2=C2+Dlog(I)*C0
12940         LET C3=C3+Dlog(I)*Dlog(I)*C0
12950         LET K=K+1!Tally another non-zero term
12960       END IF
12970     NEXT I
12980     IF C1<>CMPLX(0,0) AND K>1 THEN ! Log func deriv
12990       REM 0th, 1st, & 2nd logarithmic derivs
13000       LET C2=C2/C1
13010       LET C3=C3/C1-C2*C2
13020       LET C1=LOG(C1)
13030       REM Newton-Raphson estimate via 2nd degree polynomial
13040       LET K1=SGN(REAL(C1))
13050       LET K2=SGN(REAL(C2))
13060       LET C0=CMPLX(0,0)
13070       SELECT K2
13080       CASE 0
13090      !PRINT " o";
13100         IF C3<>CMPLX(0,0) THEN LET C0=-2*C1/C3
13110         IF C0<>CMPLX(0,0) THEN LET Alf=Alf+K1*SQR(C0)
13120       CASE Lsn
```

```
13130      !PRINT " +";
13140         LET C0=C2*C2-2*C1*C3
13150         IF C0=CMPLX(0,0) THEN
13160           LET C0=2*C1/C2
13170         ELSE
13180           LET C0=2*C1/(C2+K2*SQR(C0))
13190         END IF
13200         LET Alf=Alf-C0   !New estm of exp factor
13210       CASE -Lsn
13220      !PRINT " -";
13230         LET C0=C2*C2-2*C1*C3
13240         IF C0=CMPLX(0,0) THEN
13250           IF C3<>CMPLX(0,0) THEN LET C0=C2/C3
13260         ELSE
13270           IF C3<>CMPLX(0,0) THEN C0=(C2+K2*SQR(C0))/C3
13280         END IF
13290         LET Alf=Alf-C0   !New estm of exp factor
13300       END SELECT
13310       LET Alferr=ABS(REAL(Alf)-REAL(Alf0))+ABS(IMAG(Alf)-IMAG(Alf0))
13320       IF ABS(REAL(Alf))>700 THEN Alf=CMPLX(Lsn/2,0)-Clg/Clg2/(1+J)!Retry
13330       LET New=1                         !Set for update
13340     END IF
13350    !PRINT " Guess";J;" ";CHR$(224);" = ";Alf;" varied~";Alferr
13360     LET J=J+1
13370   END WHILE
13380   LET Alferr=ABS(Alf0-Alf)   !Iteration variance
13390  !IF Alferr<>0 THEN !PRINT "  Iteration variance =";Alferr
13400   LET Relay(6)=REAL(Alf)     ! Relay the exp ave factor = relay(6&7)
13410   LET Relay(7)=IMAG(Alf)
13420   RETURN Alf
13430 FNEND
13440 !   [     ]    [     ]    SUB Pixs_by_hand    [     ]    [     ]    [     ]
13450 SUB Pixs_by_hand
13460 !***> Subprogram to hand fill a 3D pixel array
13470   COM /Pass/Relay(0:7)
13480   COM /Pixel/ INTEGER Xmax,Ymax,Zmax,Pixl(1:8000),REAL Xscl,Yscl,Zscl
13490   INTEGER Xhnd,Yhnd,Zhnd,Nhnd
13500   FOR Xhnd=1 TO Xmax
13510     FOR Zhnd=Zmax TO 1 STEP -1
13520       FOR Yhnd=1 TO Ymax
13530         LET Nhnd=Xhnd+(Yhnd-1+(Zhnd-1)*Ymax)*Xmax
13540         DISP "Pixl(";VAL$(Xhnd);",";VAL$(Yhnd);",";VAL$(Zhnd);")";
13550         DISP "? (previous=";VAL$(Pixl(Nhnd));")";
13560         INPUT " ",Pixl(Nhnd)
13570       NEXT Yhnd
13580     NEXT Zhnd
13590   NEXT Xhnd
13600 SUBEND
13610 !   [     ]    [     ]    [     ]    [     ]    [     ]    [     ]    [     ]
13620 SUB Pixs_by_random(INTEGER Spclmt)
13630 !***> Subprogram to create a 3D pixel array of random shuffle
13640   COM /Pass/Relay(0:7)
13650   COM /Pixel/ INTEGER Xmax,Ymax,Zmax,Pixl(1:8000),REAL Xscl,Yscl,Zscl
13660   INTEGER Xrdm,Yrdm,Zrdm,Nrdm,Nsf,Spcin,Blks,Seed,Xsf,Ysf,Zsf,Vsf
13670   INTEGER Lsf,Xvr,Yvr,Zvr
```

```
13680    REAL Vrdm,Vtot,Vlm(1:9)
13690    ! Spclmt is integer for max # of species
13700    ! COM /Pass/ uses Relay(0) as vol[1] & Relay(1) seed
13710    !  Relay(2) as random along axis selector
13720    LET Spcin=Spclmt                    !initialize species limit
13730    LET Vtot=0                          !Relative volume accumulator
13740    IF Relay(1)=0 THEN                  !uses Relay(1) as random seed
13750      DISP
13760      INPUT "Random seed? (neg to timer) ",Seed
13770      LET Xsf=1
13780      INPUT "Shuffle along the X direction? 0=no 1=yes (default=1) ",Xsf
13790      IF Xsf<0 THEN STOP
13800      LET Xsf=(Xsf>0)
13810      LET Ysf=1
13820      INPUT "Shuffle along the Y direction? 0=no 1=yes (default=1) ",Ysf
13830      IF Ysf<0 THEN STOP
13840      LET Ysf=(Ysf>0)
13850      LET Zsf=1
13860      INPUT "Shuffle along the Z direction? 0=no 1=yes (default=1) ",Zsf
13870      IF Zsf<0 THEN STOP
13880      LET Zsf=(Zsf>0)
13890    ELSE
13900      LET Seed=INT(.000001+Relay(1))
13910      LET Vsf=INT(.5+Relay(2))          ! Vsf temporary axis selection
13920      LET Xsf=(BIT(Vsf,2)=0)            ! 1,0 for BIT(_,2)=0,1
13930      LET Ysf=(BIT(Vsf,1)=0)            ! 1,0 for BIT(_,1)=0,1
13940      LET Zsf=(BIT(Vsf,0)=0)            ! 1,0 for BIT(_,0)=0,1
13950    END IF
13960    LET Lsf=(Xsf=0 AND Ysf=0 AND Zsf=0) !special case, no shuffling
13970    IF Relay(1)<2 THEN
13980      PRINT "Subprogram ""Pixs_by_random"" fills a pixel box";Xmax;
13990      PRINT "x";Ymax;"x";Zmax;"randomly."
14000      IF Lsf THEN PRINT " NO shuffles, ";
14010      PRINT " Seed=";VAL$(Seed);"; shuffles, X's=";VAL$(Xsf);
14020      PRINT ", Y's=";VAL$(Ysf);", & Z's=";VAL$(Zsf);"."
14030    END IF
14040    LET Xsf=Xsf*(Xmax-1)+1              ! either 1 or Xmax value
14050    LET Ysf=Ysf*(Ymax-1)+1              ! either 1 or Ymax value
14060    LET Zsf=Zsf*(Zmax-1)+1              ! either 1 or Zmax value
14070    LET Vsf=Xsf*Ysf*Zsf                 ! # of elements to shuffle
14080    IF Lsf THEN Vsf=Xmax*Ymax*Zmax
14090    LET Xsf=(Xsf=1)*(Xmax-1)+1          ! either Xmax or 1 value, revrsl
14100    LET Ysf=(Ysf=1)*(Ymax-1)+1          ! either Ymax or 1 value, revrsl
14110    LET Zsf=(Zsf=1)*(Zmax-1)+1          ! either Zmax or 1 value, revrsl
14120    IF Vsf=1 THEN PRINT "TRIVAL Pixl, random shuffling along no direction!"
14130    IF Seed<0 THEN RANDOMIZE TIMEDATE MOD 32767
14140    IF Seed>0 THEN RANDOMIZE Seed
14150    FOR Nrdm=1 TO Spcin
14160      IF Spclmt=Spcin THEN
14170        IF Relay(1)>0 THEN              !use Relay(0) as volume fraction
14180          IF Nrdm=1 THEN LET Vrdm=FRACT(Relay(0))
14190          IF Nrdm=2 THEN LET Vrdm=1-FRACT(Relay(0))
14200          IF Nrdm=3 THEN LET Vrdm=-1
14210        ELSE
14220          DISP "Give occupation for pixel species [";VAL$(Nrdm);"]";
```

```
14230          DISP " of [";VAL$(Spclmt);"]? ";
14240          INPUT "(or <0 ends sequence) ",Vrdm
14250        END IF
14260        IF Vrdm<0 THEN
14270          LET Spclmt=Nrdm-1
14280        ELSE
14290          LET Vtot=Vtot+Vrdm
14300        END IF
14310      END IF
14320      IF Nrdm<=Spclmt THEN
14330        LET Vlm(Nrdm)=Vtot
14340      ELSE
14350        LET Vlm(Nrdm)=0
14360      END IF
14370    NEXT Nrdm
14380    FOR Nrdm=1 TO Spclmt
14390      IF Vtot=0 AND Spclmt>0 THEN
14400        Vlm(Nrdm)=INT(.5+Vsf*Nrdm/Spclmt)
14410      ELSE
14420        Vlm(Nrdm)=INT(.5+Vlm(Nrdm)*Vsf/Vtot)
14430      END IF
14440     !PRINT "v";VAL$(INT(Vlm(Nrdm))); !debugger
14450    NEXT Nrdm
14460    LET Blks=1                       !keep track of blocks used
14470    FOR Nrdm=1 TO Vsf                !dispense Pixels as occup. specs
14480      WHILE Nrdm>Vlm(Blks) AND Blks<Spclmt
14490        LET Blks=Blks+1              !when level exhausted move on
14500      END WHILE
14510      LET Pixl(Nrdm)=Blks
14520    NEXT Nrdm
14530    IF NOT (Lsf) THEN                !ie, normal shuffling
14540      FOR Nrdm=1 TO Vsf              !shuffle Pixels
14550        REPEAT
14560          LET Nsf=INT(RND*Vsf)
14570        UNTIL Nsf<>Nrdm AND Nsf<Vsf
14580        LET Blks=Pixl(Nrdm)          !swap
14590        LET Pixl(Nrdm)=Pixl(Nsf+1)
14600        LET Pixl(Nsf+1)=Blks
14610     ! PRINT "s";VAL$(Blks);        !!SHUFFLing sequence,debugger
14620      NEXT Nrdm!!SHUFFLE sequence
14630    ! PRINT
14640      IF NOT (Xsf=1 AND Ysf=1 AND Zsf=1) THEN !if partial, copy out
14650        LET Nsf=Vsf
14660        FOR Zrdm=Zmax TO Zsf STEP -1
14670          FOR Yrdm=Ymax TO Ysf STEP -1
14680            FOR Xrdm=Xmax TO Xsf STEP -1
14690              LET Nrdm=FNOde_mk(Xrdm,Yrdm,Zrdm,Xmax,Ymax,Zmax)
14700              LET Pixl(Nrdm)=Pixl(Nsf)
14710              LET Pixl(Nsf)=0
14720              LET Nsf=Nsf-1
14730            NEXT Xrdm
14740          NEXT Yrdm
14750        NEXT Zrdm
14760      END IF                         ! end if not entire Pixl array
14770      FOR Zrdm=1 TO Zmax
```

```
14780       LET Zvr=Zsf
14790       IF Zvr=1 THEN Zvr=Zrdm
14800       FOR Yrdm=1 TO Ymax
14810         LET Yvr=Ysf
14820         IF Yvr=1 THEN Yvr=Yrdm
14830         FOR Xrdm=1 TO Xmax
14840           Nrdm=FNOde_mk(Xrdm,Yrdm,Zrdm,Xmax,Ymax,Zmax)
14850           LET Xvr=Xsf
14860           IF Xvr=1 THEN LET Xvr=Xrdm
14870           LET Nsf=FNOde_mk(Xvr,Yvr,Zvr,Xmax,Ymax,Zmax)
14880           Pixl(Nrdm)=Pixl(Nsf)
14890         NEXT Xrdm
14900       NEXT Yrdm
14910     NEXT Zrdm
14920   END IF                            !end if, NOT(Lsf)=shuffle OK
14930 SUBEND
14940 !  [    ]    [    ]    [    ]    [    ]    [    ]    [    ]    [    ]
14950 SUB Pixs_by_2block(INTEGER Styps)
14960 !***> Subprogram to create a 3D pixel array of fractal or scaless
14970 !***> grain sizing
14980   COM /Pass/Relay(0:7)
14990   COM /Pixel/ INTEGER Xmax,Ymax,Zmax,Pixl(1:8000),REAL Xscl,Yscl,Zscl
15000   INTEGER Nran,Mran,Nwth,Styp0,Cell,Qnza,Bsz,Jftl,Kftl,Lftl,Nftl,Vftl
15010   INTEGER Aftl,Bftl,Krc,Kqc,Shldr,Xftl,Yftl,Zftl,Xpmx,Ypmx,Zpmx
15020   INTEGER Glv,Myr,Cycl
15030   REAL Vsth,Vall,Vact(1:9)
15040   ! Styps is integer for max # of species, may be changed by SUB
15050   ! COM /Pass/ uses Relay(0) as vol[1] & Relay(1) seed
15060   LET Styp0=Styps                  !initialize species limit
15070   LET Vftl=SHIFT(Xmax,1)*SHIFT(Ymax,1)*SHIFT(Zmax,1) !#2x2x2s
15080   LET Jftl=Xmax*Ymax               !pixels on a Z level
15090   LET Xpmx=Xmax-1                  !next to maximums
15100   LET Ypmx=Ymax-1
15110   LET Zpmx=Zmax-1
15120   LET Nftl=SHIFT(Vftl,1)           !about 50%, 2x2x2s
15130   ALLOCATE INTEGER Sqnc(1:Nftl)    !array of 2x2x2 locations
15140   LET Vall=0                       !Relative volume accumulator
15150   IF Relay(1)=0 THEN               !uses Relay(1) as random seed
15160     DISP
15170     INPUT "Random seed? (neg to timer) ",Qnza
15180   ELSE
15190     LET Qnza=INT(.000001+Relay(1))
15200   END IF
15210   IF Relay(1)<2 THEN
15220     PRINT "Subprogram ""Pixs_by_2block"" fills a pixel box";Xmax;
15230     PRINT "x";Ymax;"x";Zmax;"randomly."
15240     PRINT " shuffle addr: ";
15250   END IF
15260   LET Bsz=Xmax*Ymax*Zmax
15270   IF Nftl<2 OR Vftl=0 THEN           !if Pixl box small
15280     FOR Nran=1 TO Bsz
15290       Pixl(Nran)=BIT(Nran,0)
15300     NEXT Nran
15310     PRINT " Pixel box too small to fractalize, try regular RANDOM"
15320   ELSE
```

```
15330     IF Bsz=1 THEN PRINT "TRIVAL Pixl, random shuffling along no direction!"
15340     IF Qnza<0 THEN RANDOMIZE TIMEDATE MOD 32767
15350     IF Qnza>0 THEN RANDOMIZE Qnza
15360     FOR Nran=1 TO StypO
15370       IF Styps=StypO THEN
15380         IF Relay(1)>0 THEN         !use Relay(0) as volume fraction
15390           IF Nran=1 THEN LET Vsth=FRACT(Relay(0))
15400           IF Nran=2 THEN LET Vsth=1-FRACT(Relay(0))
15410           IF Nran=3 THEN LET Vsth=-1
15420         ELSE
15430           DISP "Give occupation for pixel species [";VAL$(Nran);"]";
15440           DISP " of [";VAL$(Styps);"]? ";
15450           INPUT "(or <0 ends sequence) ",Vsth
15460         END IF
15470         IF Vsth<0 THEN
15480           LET Styps=Nran-1
15490         ELSE
15500           LET Vall=Vall+Vsth
15510         END IF
15520       END IF
15530       IF Nran<=Styps THEN
15540         LET Vact(Nran)=Vall
15550       ELSE
15560         LET Vact(Nran)=0
15570       END IF
15580     NEXT Nran
15590     FOR Nran=1 TO Styps
15600       IF Vall=0 AND Styps>0 THEN
15610         Vact(Nran)=INT(.5+Bsz*Nran/Styps)
15620       ELSE
15630         Vact(Nran)=INT(.5+Vact(Nran)*Bsz/Vall)
15640       END IF
15650    !PRINT "v";VAL$(INT(Vact(Nran))); !debugger
15660     NEXT Nran
15670     LET Cell=1                      !keep track of blocks used
15680     LET Kftl=0                      ! count Pixl=[1]s
15690     FOR Nran=1 TO Bsz               !dispense Pixels as occup. specs
15700       WHILE Nran>Vact(Cell) AND Cell<Styps
15710         LET Cell=Cell+1             !when level exhausted move on
15720       END WHILE
15730       LET Pixl(Nran)=Cell
15740       IF Cell=1 THEN Kftl=Kftl+1    ! increment count of Pixl=[1]s
15750     NEXT Nran
15760     FOR Nran=1 TO Nftl              ! make 2x2x2 locations
15770       REPEAT
15780         LET Shldr=1
15790         REPEAT                      ! get randomly X up to Xmax-1, etc
15800           LET Xftl=INT(RND*Xpmx)
15810         UNTIL Xftl<Xpmx
15820         REPEAT
15830           LET Yftl=INT(RND*Ypmx)
15840         UNTIL Yftl<Ypmx
15850         REPEAT
15860           LET Zftl=INT(RND*Zpmx)
15870         UNTIL Zftl<Zpmx
```

```
15880         LET Nwth=1+Xftl+Yftl*Xmax+Zftl*Jftl !formulate Pixl address
15890         FOR Mran=1 TO Nran-1        ! check if good new address
15900           LET Aftl=Nwth-Sqnc(Mran) ! address differences
15910           IF Aftl=0 THEN            ! at same location
15920             LET Shldr=0
15930           ELSE                      ! or neighbor checking
15940             LET Zftl=ABS(Aftl DIV Jftl)
15950             LET Xftl=ABS(Aftl MOD Jftl)
15960             LET Yftl=ABS(Xftl DIV Xmax)
15970             LET Xftl=ABS(Xftl MOD Xmax)
15980             LET Bftl=(Xftl>1 OR Yftl>1 OR Zftl>1) !test, not overlap
15990            !PRINT VAL$(Bftl);
16000             IF NOT Bftl THEN Shldr=0
16010           END IF                    !endif, neighbor checking, Aftl
16020         NEXT Mran                   !next checking in sequence
16030         IF Shldr THEN Sqnc(Nran)=Nwth !assign another 2x2x2
16040       UNTIL Shldr
16050     NEXT Nran
16060     FOR Nran=1 TO Nftl-1            !shuffle 2x2x2s
16070       LET Mran=Sqnc(Nran)           !randomly exchange in sequence
16080       REPEAT                        !with another higher up
16090         LET Nwth=INT(RND*(Nftl-Nran))
16100       UNTIL Nwth<(Nftl-Nran)
16110       LET Nwth=Sqnc(Nwth+Nran+1)
16120       IF Relay(1)<2 THEN PRINT Mran;"<=>";Nwth;";"; !exchange details
16130       FOR Krc=0 TO 7                !range thru Pixls in 2x2x2s
16140         LET Aftl=BIT(Krc,2)+BIT(Krc,1)*Xmax+BIT(Krc,0)*Jftl
16150         LET Cell=Pixl(Mran+Aftl)
16160         LET Pixl(Mran+Aftl)=Pixl(Nwth+Aftl)
16170         LET Pixl(Nwth+Aftl)=Cell
16180       NEXT Krc
16190     NEXT Nran
16200     IF Relay(1)<2 THEN PRINT
16210     FOR Nran=1 TO Nftl              ! add random orientations
16220       LET Mran=Sqnc(Nran)
16230       REPEAT                        ! get a random #
16240         LET Krc=INT(RND*16)
16250       UNTIL Krc<16
16260       LET Kqc=SHIFT(Krc,2)          ! 0=none 1<>Xs 2<>Ys 3<>Zs
16270       IF Kqc>0 THEN                 ! Kqc>0 >=> random exchanges
16280         FOR Glv=0 TO 1              ! do 2 levels the same
16290           FOR Myr=0 TO BIT(Krc,1)   ! 0=diag 1=flatwise (2 passes)
16300             LET Xftl=Glv            ! 1st exchange relv coordinates
16310             LET Yftl=Myr
16320             LET Zftl=BIT(Krc,0)
16330             IF Myr THEN Zftl= NOT Zftl! reverse 2nd pass of Myr
16340             FOR Cycl=2 TO Kqc       ! cyclic permutation
16350               LET Cell=Zftl
16360               LET Zftl=Yftl
16370               LET Yftl=Xftl
16380               LET Xftl=Cell
16390             NEXT Cycl
16400             LET Aftl=1+Xftl+Yftl*Xmax+Zftl*Jftl
16410            !PRINT " <";VAL$(Mran);">(";VAL$(Xftl);",";VAL$(Yftl);",";
16420            !PRINT VAL$(Zftl);")< >";
```

```
16430                LET Xftl=Glv          ! 2nd exchange coordinates
16440                LET Yftl=Myr          ! same as 1st but add NOTs
16450                LET Zftl=BIT(Krc,0)
16460                IF Myr THEN Zftl= NOT Zftl! reverse upon 2nd pass
16470                IF BIT(Krc,0)=0 OR BIT(Krc,1)=0 THEN Yftl= NOT Yftl
16480                IF BIT(Krc,0)=1 OR BIT(Krc,1)=0 THEN Zftl= NOT Zftl
16490                FOR Cycl=2 TO Kqc      ! cyclic permutation
16500                  LET Cell=Zftl
16510                  LET Zftl=Yftl
16520                  LET Yftl=Xftl
16530                  LET Xftl=Cell
16540                NEXT Cycl
16550                LET Bftl=1+Xftl+Yftl*Xmax+Zftl*Jftl
16560               !PRINT "(";VAL$(Xftl);",";VAL$(Yftl);",";VAL$(Zftl);")";
16570                LET Cell=Pixl(Mran+Aftl)! swap
16580                LET Pixl(Mran+Aftl)=Pixl(Mran+Bftl)
16590                LET Pixl(Mran+Bftl)=Cell
16600              NEXT Myr
16610            NEXT Glv
16620          END IF                       ! endif, Kqc>0, random exchanges
16630          FOR Krc=0 TO 7               ! tag the large blocks
16640            LET Aftl=1+BIT(Krc,2)+BIT(Krc,1)*Xmax+BIT(Krc,0)*Jftl
16650            LET Pixl(Mran+Aftl)=BINCMP(Pixl(Mran+Aftl))
16660          NEXT Krc
16670        NEXT Nran
16680        IF (Bsz-SHIFT(Nftl,-3))>2 THEN ! shuffle rest unmarked
16690          FOR Nran=1 TO Bsz
16700            IF Pixl(Nran)>=0 THEN
16710              REPEAT
16720                LET Mran=Bsz*RND
16730                IF Mran<Bsz THEN       ! test for Pixl marked/unmarked
16740                  LET Myr=Pixl(Mran+1)
16750                ELSE
16760                  LET Myr=-1
16770                END IF
16780              UNTIL Mran<>Nran AND Myr>=0
16790              LET Mran=Mran+1
16800              LET Cell=Pixl(Nran)      ! swap
16810              LET Pixl(Nran)=Pixl(Mran)
16820              LET Pixl(Mran)=Cell
16830            END IF                     ! end if, Pixl test
16840          NEXT Nran
16850        END IF                         ! end if, shuffle unmarked
16860        LET Cell=0                     ! start to count Pixl=[1]s
16870        FOR Nran=1 TO Bsz              ! remove marking
16880          IF Pixl(Nran)<0 THEN Pixl(Nran)=BINCMP(Pixl(Nran))
16890          IF Pixl(Nran)=1 THEN Cell=Cell+1
16900        NEXT Nran
16910        IF Cell<>Kftl THEN PRINT " Lost/gained something in SUB shuffle"
16920       !PRINT
16930       !PRINT Sqnc(*)
16940        DEALLOCATE Sqnc(*)
16950      END IF                           !end if, for big enuf Pixl box
16960 SUBEND
16970 !    |    |    |    |    |    |    |    |    |    |    |    |    |    |
```

```
16980 SUB Pixs_by_ellps
16990   COM /Pass/Relay(0:7)
17000   COM /Pixel/ INTEGER Xmax,Ymax,Zmax,Pixl(1:8000),REAL Xscl,Yscl,Zscl
17010   !***> Creates a 3D pixel array of an ellisoidal inclusion in a host
17020   INTEGER Blkc1,Blkc2,Boxz,Btst,Dsr,Impv,Roll,Tally,Uyou,Xlv,Ylv,Zlv
17030   REAL Chgu,Flvr,Frs1,Frs2,Guess
17040   REAL Xgu,Ygu,Zgu,Vlps,Xlps,Ylps,Zlps,Xflv,Yflv
17050   LET Boxz=Xmax*Ymax*Zmax
17060   IF Boxz<=0 THEN PRINT Xmax;"x";Ymax;"x";Zmax;"??? in SUB ellps"
17070   LET Uyou=(Relay(0)=0)              !user input signal
17080   IF Uyou THEN
17090     DISP
17100     PRINT "Filling pixels with ellipsoid inclusion imbedded in";
17110     PRINT " a host (a binary)"
17120     LET Frs1=1
17130     IF Relay(0)>0 THEN Frs1=Relay(0)
17140     DISP "Give occupation by species [1]? (default=";VAL$(Frs1);
17150     INPUT ") ",Frs1
17160     LET Frs2=1
17170     IF Frs1>0 AND Frs1<1 THEN Frs2=1-Frs1
17180     DISP "Give occupation by species [2]? (default=";VAL$(Frs2);
17190     INPUT ") ",Frs2
17200     DISP "Inclusion role? 0)=smaller species, 1)=species #1,";
17210     INPUT " 2)=species #2 ",Roll
17220     IF Roll<0 THEN STOP
17230     LET Xlps=1
17240     INPUT "Ellipsoid X axis length? (default=1) ",Xlps
17250     LET Ylps=1
17260     INPUT "Ellipsoid Y axis length? (default=1) ",Ylps
17270     LET Zlps=1
17280     INPUT "Ellipsoid Z axis length? (default=1) ",Zlps
17290     PRINT "Subprogram ""Pixs_by_ellps"" fills a pixel box";Xmax;
17300     PRINT "x";Ymax;"x";Zmax;"with an ellipsiod"
17310   ELSE
17320     LET Frs1=FRACT(Relay(0))        !Relay(0)=volume fraction of [1]
17330     LET Frs2=1-Frs1
17340     LET Roll=(Relay(4)>.5)          !Relay(4) is role reversal flag
17350     LET Xlps=Relay(1)               !Relay(1) is X ellipse axis
17360     LET Ylps=Relay(2)               !Relay(2) is Y ellipse axis
17370     LET Zlps=Relay(3)               !Relay(3) is Z ellipse axis
17380   END IF                            !end if,Uyou
17390   IF Roll>2 THEN LET Roll=2-BIT(Roll,0)
17400   LET Frs2=Frs1+Frs2
17410   IF Frs2>0 THEN Frs1=Frs1/Frs2     !Normalize fractions
17420   LET Frs2=1-Frs1
17430   IF Roll=0 THEN Roll=1+(Frs1>.5)  !program selects inclusion role
17440   IF Roll=1 THEN Dsr=INT(.5+Boxz*Frs1)
17450   IF Roll=2 THEN Dsr=INT(.5+Boxz*Frs2)
17460   IF Xlps<=0 THEN Xlps=1
17470   IF Ylps<=0 THEN Ylps=1
17480   IF Zlps<=0 THEN Zlps=1
17490   LET Vlps=.5*PI*Xlps*Ylps*Zlps     !3*ellipse quandrant user size param
17500   LET Impv=0
17510   LET Guess=(3*Dsr/Vlps)^(1/3)      !initial guess at axis scaling factor
17520   LET Blkc1=SHIFT(-1,1)             !assume worst case to start,MAXINT
```

```
17530   LET Blkc2=SHIFT(-1,1)              !assume worst case to start,MAXINT
17540   LET Impv=4                         !least 3 guesses
17550   REPEAT                             !until guess improved
17560     LET Impv=Impv-1
17570     LET Xgu=Guess*Xlps
17580     LET Ygu=Guess*Ylps
17590     LET Zgu=Guess*Zlps
17600     LET Xgu=Xgu*Xgu
17610     LET Ygu=Ygu*Ygu
17620     LET Zgu=Zgu*Zgu
17630     LET Tally=0
17640     FOR Xlv=1 TO Xmax
17650       LET Xflv=((.5+Xlv)*(.5+Xlv))/Xgu
17660       FOR Ylv=1 TO Ymax
17670         LET Yflv=((.5+Ylv)*(.5+Ylv))/Ygu
17680         FOR Zlv=1 TO Zmax
17690           LET Flvr=((.5+Zlv)*(.5+Zlv))/Zgu+Xflv+Yflv
17700           LET Tally=Tally+(Flvr<=1) !Tally pixels within ellipsoid
17710         NEXT Zlv
17720       NEXT Ylv
17730     NEXT Xlv
17740     LET Btst=Tally-Dsr                !How far off the mark is the Tally
17750    !PRINT "c";VAL$(Btst);
17760     IF Btst<0 THEN
17770       IF -Btst<Blkc1 THEN             !new lower minumum found
17780         LET Blkc1=-Btst
17790         LET Impv=3
17800       END IF
17810     END IF
17820     IF Btst>0 THEN                    !new upper maximum found
17830       IF Btst<Blkc2 THEN
17840         LET Blkc2=Btst
17850         LET Impv=3
17860       END IF
17870     END IF
17880     LET Chgu=Btst/(Guess*Guess*Vlps) !estimate correction to guess
17890     IF Impv>0 THEN Guess=Guess-Chgu
17900     IF Guess<=0 AND Impv>0 THEN Guess=(Guess+Chgu)*.5
17910   UNTIL Impv<=0                       !until nolonger improves
17920  !PRINT
17930   IF Guess<>0 THEN
17940     IF Guess<0 THEN Guess=-Guess
17950     FOR Xlv=1 TO Xmax
17960       LET Xflv=((.5+Xlv)*(.5+Xlv))/Xgu
17970       FOR Ylv=1 TO Ymax              !assign Pixls
17980         LET Yflv=((.5+Ylv)*(.5+Ylv))/Ygu
17990         FOR Zlv=1 TO Zmax
18000           LET Flvr=((.5+Zlv)*(.5+Zlv))/Zgu+Xflv+Yflv
18010           LET Tally=Xlv+(Ylv-1+(Zlv-1)*Ymax)*Xmax
18020           IF Roll=1 THEN LET Pixl(Tally)=1+(Flvr>1)
18030           IF Roll=2 THEN LET Pixl(Tally)=2-(Flvr>1)
18040         NEXT Zlv
18050       NEXT Ylv
18060     NEXT Xlv
18070     IF Uyou THEN
```

```
18080        PRINT " whose size is: X axis=";VAL$(DROUND(Xlps*Guess,4));
18090        PRINT ", Y axis=";VAL$(DROUND(Ylps*Guess,4));
18100        PRINT ", & Z axis=";VAL$(DROUND(Zlps*Guess,4));"."
18110      END IF
18120    END IF                             !end if guess<>0
18130 SUBEND
18140 SUB Pix_by_correlat
18150    COM /Pass/Relay(0:7)
18160    COM /Pixel/ INTEGER Xmax,Ymax,Zmax,Pixl(1:8000),REAL Xscl,Yscl,Zscl
18170    INTEGER Bcrl,Crrl,Gcrl,Icrl,Jcrl,Kcrl,Lcrl,Ncrl,Qcrl,Scrl,Tcrl,Zcrl
18180    INTEGER Xc,Yc,Zc,Xdf,Ydf,Zdf
18190    REAL Arel,Prel,Rrel,Vrel,Vsm,Frel(1:9)
18200    LET Zcrl=Xmax*Ymax                 ! Pixel box area of Z section
18210    LET Bcrl=Zcrl*Zmax                 ! Total max Pixels
18220    ALLOCATE Tabl(1:Bcrl),Wrel(0:Bcrl) ! For correlation weighting
18230    MAT Pixl=(0)                       ! Need to zero out pixels
18240    LET Vsm=0                          ! Volume fraction
18250    LET Crrl=INT(Relay(2))             ! Pass correlation selection
18260    IF Crrl=1 OR Crrl=4 THEN Prel=Relay(3) ! Pass correlation length
18270    IF Relay(1)=0 THEN
18280      DISP
18290      INPUT "Random seed? (negative to timer)",Scrl
18300      DISP "Correlation? none=0 expon=1 inverse=2 inv square=3";
18310      DISP " power law=4 (default=";VAL$(Crrl);
18320      INPUT ")",Crrl
18330      LET Relay(2)=Crrl
18340    ELSE
18350      LET Scrl=(.00000000001+Relay(1))
18360    END IF
18370    IF Scrl>0 THEN RANDOMIZE Scrl    ! Seeding random generator
18380    IF Scrl<0 THEN RANDOMIZE TIMEDATE MOD 32767
18390    LET Vrel=(Xmax-1)*(Xmax-1)+(Ymax-1)*(Ymax-1)+(Zmax-1)*(Zmax-1)
18400     !LET Vrel=Xmax*Xmax+Ymax*Ymax+Zmax*Zmax ! =hypotenuse diag squared
18410    SELECT Crrl
18420    CASE =1                            ! exponential correlation
18430      IF Relay(1)=0 THEN
18440        DISP "Correlation length? (in pixel units, default=";
18450        DISP VAL$(Prel);
18460        INPUT ")",Prel
18470      END IF
18480      IF Prel=0 THEN Prel=SQR(Vrel)*.5   ! if zero, then .5 diag
18490      LET Relay(3)=Prel
18500      LET Arel=EXP(SQR(Vrel)/Prel)     ! field weighting factor
18510    CASE =2                            ! inverse correlation
18520      LET Arel=SQR(Vrel)
18530    CASE =3                            ! inverse square correlation
18540      LET Arel=Vrel
18550    CASE =4                            ! power law correlation
18560      IF Relay(1)=0 THEN
18570        DISP "Power law? (1=inverse, 2=inv sqr, etc, default=";
18580        DISP VAL$(Prel);
18590        INPUT ")",Prel
18600        LET Relay(3)=Prel
18610      END IF
18620      IF Prel=0 THEN
```

```
18630       LET Crrl=0
18640     ELSE
18650       LET Arel=Vrel^(Prel*.5)
18660     END IF
18670   CASE ELSE
18680     LET Arel=1
18690   END SELECT
18700   LET Tcrl=0                         ! Species tally
18710   FOR Ncrl=1 TO 9                    ! up to 9 species
18720     LET Frel(Ncrl)=0
18730     IF Tcrl=0 THEN                   ! Has not ended sequence
18740       IF Relay(1)>0 THEN             ! get next occupation
18750         IF Ncrl=1 THEN Vrel=FRACT(Relay(0))
18760         IF Ncrl=2 THEN Vrel=1-FRACT(Relay(0))
18770         IF Ncrl=3 THEN Vrel=-1
18780       ELSE                           ! manual operation
18790         DISP "Give occupation for species [";VAL$(Ncrl);
18800         INPUT "] or (<0 ends sequence)",Vrel
18810       END IF
18820       IF Vrel<0 THEN
18830         LET Tcrl=Ncrl-1              ! # of species at sequence end
18840         LET Frel(Ncrl)=0
18850       ELSE
18860         LET Vsm=Vsm+Vrel
18870         LET Frel(Ncrl)=Vsm           ! Accumulative volume
18880       END IF
18890     END IF
18900   NEXT Ncrl
18910   IF Vsm<>0 THEN LET Vsm=Bcrl/Vsm    ! Normalize to Pixel allotment
18920   FOR Ncrl=1 TO Tcrl                 ! Intergerize up to Bcrl total
18930     IF Frel(Ncrl)<>0 THEN Frel(Ncrl)=INT(Frel(Ncrl)*Vsm+.5)
18940     IF Vsm=0 THEN Frel(Ncrl)=INT(Ncrl*Bcrl/Tcrl+.5)
18950   NEXT Ncrl
18960   FOR Ncrl=Tcrl TO 2 STEP -1         ! Convert to allotments
18970     LET Frel(Ncrl)=INT(Frel(Ncrl)-Frel(Ncrl-1)+.5)
18980   NEXT Ncrl
18990   LET Vsm=SUM(Frel)
19000   IF Vsm<>Bcrl THEN PRINT "Unable to place all pixels, see SUB"
19010   IF Relay(1)=0 THEN
19020     PRINT " pixs allocations, sum=";VAL$(Vsm);" @";
19030     PRINT Frel(*);
19040     PRINT
19050   END IF
19060   LET Xc=0                           ! Initialize addresses
19070   LET Yc=1
19080   LET Zc=1
19090   FOR Ncrl=1 TO Bcrl                 ! Set up weights tabulation
19100     LET Xc=Xc+1                      ! to avoid excessive recomputations
19110     IF Xc>Xmax THEN
19120       LET Xc=1
19130       LET Yc=Yc+1
19140       IF Yc>Ymax THEN
19150         LET Yc=1
19160         LET Zc=Zc+1
19170       END IF
```

```
19180      END IF
19190      LET Vrel=(Xc-1)*(Xc-1)+(Yc-1)*(Yc-1)+(Zc-1)*(Zc-1) ! sep distance
19200      SELECT Crrl                          ! select correlation potentials
19210      CASE =0
19220        LET Vrel=1
19230      CASE =1
19240        LET Vrel=EXP(SQR(Vrel)/Prel)
19250      CASE =2
19260        IF Vrel<>0 THEN Vrel=SQR(Vrel)
19270      CASE =4
19280        IF Vrel<>0 THEN Vrel=Vrel^(Prel*.5)
19290      END SELECT
19300      IF Vrel=0 THEN
19310        LET Tabl(Ncrl)=0
19320      ELSE
19330        LET Tabl(Ncrl)=Arel/Vrel
19340      END IF
19350    NEXT Ncrl
19360    IF Relay(1)=0 THEN                    ! correlation potentials map
19370      FOR Xc=1 TO Xmax
19380        PRINT "X-section weight to origin, plane =";Xc
19390        FOR Zc=Zmax TO 1 STEP -1
19400          FOR Yc=1 TO Ymax
19410            LET Ncrl=Xc+(Yc-1)*Xmax+(Zc-1)*Zcrl
19420            PRINT USING """|"".7A,#";VAL$(DROUND(Tabl(Ncrl),3))
19430          NEXT Yc
19440          PRINT
19450        NEXT Zc
19460      NEXT Xc
19470    END IF                                ! end if, map
19480    REM  Finding the weighting field . . . . . . . . . . . >
19490    PRINT "  Wait filling";Bcrl;"Pixels in SUB Pix_by_correlat"
19500    LET Rrel=RND                          ! prime the RND generator
19510    FOR Icrl=Bcrl TO 1 STEP -1            ! Roam thru random fill positions
19520      LET Kcrl=0
19530      LET Gcrl=0
19540      FOR Jcrl=1 TO Tcrl                  ! Fill random ordering sequence
19550        IF INT(Frel(Jcrl))>.5 THEN
19560          LET Kcrl=Kcrl+INT(Frel(Jcrl)+.5) ! summation of remaining to fill
19570          LET Gcrl=Gcrl+1                 ! tally of remaining pixel species
19580        END IF
19590      NEXT Jcrl
19600      IF Gcrl>1 THEN                      ! Fill randomly,>1 pixels to fill
19610        REPEAT
19620          LET Ncrl=INT(RND*Kcrl)          ! get random #
19630        UNTIL Ncrl<Kcrl
19640        LET Ncrl=Ncrl+1
19650        LET Lcrl=0
19660        LET Kcrl=0
19670        FOR Jcrl=1 TO Tcrl
19680          IF Frel(Jcrl)>.5 AND Lcrl=0 THEN
19690            LET Kcrl=Kcrl+INT(Frel(Jcrl)+.5)
19700            IF Kcrl>=Ncrl THEN             ! contains address of non-zero
19710              LET Lcrl=Jcrl
19720              LET Frel(Lcrl)=Frel(Lcrl)-1! reduce one more
```

```
19730          END IF
19740        END IF
19750      NEXT Jcrl
19760      PRINT "[";VAL$(Lcrl);"]";
19770      MAT Wrel=(0)
19780      LET Kcrl=0
19790      FOR Jcrl=1 TO Bcrl             ! Find pixel weighting field
19800        LET Zc=(Jcrl-1) DIV Zcrl
19810        LET Xc=(Jcrl-1) MOD Zcrl
19820        LET Yc=Xc DIV Xmax
19830        LET Xc=Xc MOD Xmax
19840        IF Pixl(Jcrl)=0 THEN         ! for unfilled or zeroed pixels
19850          LET Kcrl=Kcrl+1
19860          FOR Ncrl=1 TO Bcrl
19870            IF Pixl(Ncrl)=Lcrl THEN ! This pixel contributes to weighting
19880              LET Zdf=ABS(((Ncrl-1) DIV Zcrl)-Zc)
19890              LET Qcrl=(Ncrl-1) MOD Zcrl
19900              LET Ydf=ABS((Qcrl DIV Xmax)-Yc)
19910              LET Xdf=ABS((Qcrl MOD Xmax)-Xc)
19920              LET Qcrl=1+Xdf+Ydf*Xmax+Zdf*Zcrl! for tabulation address
19930             !LET Vrel=Xdf*Xdf+Ydf*Ydf+Zdf*Zdf !Square of rel diff in addr
19940              IF Qcrl>0 THEN Wrel(Kcrl)=Wrel(Kcrl)+Tabl(Qcrl)
19950            END IF                   ! end if, Pixl=Lcrl test
19960          NEXT Ncrl
19970        END IF                       ! end if, Pixl=0 test
19980      NEXT Jcrl
19990      LET Vsm=0
20000      FOR Jcrl=1 TO Kcrl             ! total of weighting
20010        LET Vsm=Vsm+Wrel(Jcrl)
20020      NEXT Jcrl
20030      IF Vsm<=0 THEN
20040        FOR Jcrl=1 TO Kcrl
20050          LET Wrel(Jcrl)=1
20060        NEXT Jcrl
20070        LET Vsm=Kcrl
20080        PRINT "e";
20090      END IF
20100 !    IF Icrl<Bcrl-3 AND Icrl>Bcrl-5 THEN ! Set for intermediate printout
20110 !      PRINT
20120 !      PRINT " Weighting map at fill #";VAL$(Bcrl-Icrl+1);":"
20130 !      FOR Zdf=1 TO Zmax
20140 !        PRINT " Pixels in Z-sectional plane";Zdf;
20150 !        PRINT "(interaction to [";VAL$(Lcrl);"])"
20160 !        FOR Ydf=1 TO Ymax
20170 !          LET Jcrl=(Ydf-1)*Xmax+(Zdf-1)*Zcrl
20180 !          FOR Xdf=1 TO Xmax
20190 !            LET Qcrl=Jcrl+Xdf
20200 !            PRINT USING "2X,""["",A,""]"",3X,#";VAL$(Pixl(Qcrl))
20210 !          NEXT Xdf
20220 !          PRINT
20230 !        NEXT Ydf
20240 !      NEXT Zdf
20250 !      LET Qcrl=0
20260 !      LET Zdf=0
20270 !      FOR Jcrl=1 TO Bcrl
```

```
20280 !          IF (Jcrl-1) MOD Zcrl=0 THEN
20290 !            LET Zdf=Zdf+1
20300 !            PRINT "Weights in Z plane";Zdf
20310 !          END IF
20320 !          IF Pixl(Jcrl)=0 THEN
20330 !            LET Qcrl=Qcrl+1
20340 !            PRINT USING """|"",7A,#";VAL$(INT(Wrel(Qcrl)+.5))
20350 !          ELSE
20360 !            PRINT "|none    ";
20370 !          END IF
20380 !          IF Jcrl MOD Xmax=0 THEN PRINT
20390 !        NEXT Jcrl
20400 !      END IF                       ! end if, intermediate printout
20410        REPEAT                       ! Get a RANDOM #
20420          LET Rrel=RND*Vsm
20430        UNTIL Rrel<Vsm
20440        LET Qcrl=1                   ! Testing switch is set to ON
20450        LET Kcrl=0                   ! Keep tally of Pixels with zeros
20460        LET Vsm=0                    ! Zero weighting accumulator
20470        FOR Jcrl=1 TO Bcrl
20480          IF Qcrl AND Pixl(Jcrl)=0 THEN ! Test if Pixel is zero
20490            LET Kcrl=Kcrl+1          ! Further assign if = RANDOM #
20500            LET Vsm=Vsm+Wrel(Kcrl)   ! Accumalative weight
20510            IF Vsm>Rrel THEN         ! Trips when reach RANDOM #
20520              LET Pixl(Jcrl)=Lcrl    ! Pixel assigned
20530              LET Qcrl=0             ! No more testing set
20540            END IF
20550          END IF                     ! end if, Pixl=0 test
20560        NEXT Jcrl
20570      ELSE                           ! Only 0 OR 1 TYPES, fast fill
20580        LET Lcrl=0
20590        FOR Jcrl=1 TO Tcrl
20600          IF Frel(Jcrl)<>0 THEN Lcrl=Jcrl
20610        NEXT Jcrl
20620        IF Lcrl>0 THEN
20630          FOR Jcrl=1 TO Bcrl         ! filling the remaining, 1 type
20640            IF Pixl(Jcrl)=0 THEN
20650              LET Pixl(Jcrl)=Lcrl
20660              LET Frel(Lcrl)=Frel(Lcrl)-1
20670            END IF
20680          NEXT Jcrl
20690        END IF
20700      END IF                         ! end if, Gcrl>1 test
20710    NEXT Icrl
20720    IF Relay(1)=0 THEN PRINT "&";SUM(Frel);"remain."
20730    DEALLOCATE Tabl(*),Wrel(*)       ! Bye-bye variable sub arrays
20740 SUBEND
20750 SUB Pixs_by_evolv(INTEGER Gpcs)
20760    COM /Pass/Relay(0:7)
20770    COM /Pixel/ INTEGER Xmax,Ymax,Zmax,Pixl(1:8000),REAL Xscl,Yscl,Zscl
20780    INTEGER Cnk,Clmx,Gens,Shv,Fadr,Rlr,Mdl,Mlkl,Mlk2,Nsft,Rset,Sadr,Vlv
20790    INTEGER Gski,Kski,Mski,Qski,Sski,Xski,Yski,Zski,Ncs(0:9),Pcs(0:9)
20800    REAL Agen,Dv,Prpt,Vski,Dmd(0:9)
20810    LET Cnk=Xmax*Ymax*Zmax
20820    FOR Sski=1 TO Cnk
```

```
20830     LET Pixl(Sski)=0                   ! Need to zero out pixels
20840   NEXT Sski
20850   LET Vski=0                           ! Volume fraction
20860   LET Mski=(Relay(1)=0)                ! manual indicator
20870   LET Rset=Relay(2)                    ! Pass vols preservation
20880   LET Rset=BIT(Rset,0)
20890   LET Gens=Relay(3)                    ! Pass generation size
20900   LET Vlv=Relay(4)                     ! Evolution starter
20910   LET Prpt=FRACT(Relay(5))             ! proportion change
20920   IF Gpcs=0 THEN Gpcs=2                ! default to binary
20930   IF Mski THEN
20940     DISP
20950     INPUT "Random seed? (negative to timer, none=0)",Shv
20960     IF Shv=0 THEN                      ! From generation to generation
20970       LET Rset=1                       ! preserve vol fractions
20980     ELSE
20990       DISP "Maintain constant parent to descendent";
21000       DISP " ratios? Vary=0 Fixed=1 (default=";VAL$(Rset);
21010       INPUT ")",Rset
21020       LET Rset=BIT(Rset,0)
21030       LET Relay(2)=Rset
21040     END IF
21050     DISP "Generations of evolution? (self determine=0, default=";Gens;
21060     INPUT ")",Gens
21070     LET Relay(3)=Gens
21080   ELSE                                 ! get parameters from Relays
21090     LET Shv=Relay(1)                   ! seed
21100     IF Shv=0 THEN
21110       LET Rset=1
21120     ELSE
21130       LET Gpcs=2
21140       Dmd(1)=FRACT(Relay(0))
21150       Dmd(2)=1-Dmd(1)
21160     END IF
21170   END IF                               ! end if, relays
21180   IF Gens=0 THEN
21190     LET Agen=MAX(Xmax,Ymax,Zmax)       ! set Gens upward integer
21200     LET Gens=-INT(1-LOG(Agen)/LOG(2.0))
21210   END IF
21220   IF Gens<0 THEN STOP
21230   IF Gens=0 THEN PRINT "No evolutions?"
21240   LET Qski=7                           ! elements per expansion cube
21250   LET Mlk1=Xmax                        ! set multipiers for addresses
21260   IF Xmax=1 THEN Mlk1=Ymax
21270   LET Mlk2=Xmax*Ymax
21280   IF Xmax=1 THEN LET Qski=SHIFT(Qski,1)
21290   IF Ymax=1 THEN LET Qski=SHIFT(Qski,1)
21300   IF Zmax=1 THEN LET Qski=SHIFT(Qski,1)
21310   LET Clmx=Qski+1
21320   ALLOCATE REAL Ruls(0:Gpcs,0:9)
21330   IF Shv>0 THEN RANDOMIZE Shv          ! Seeding random generator
21340   IF Shv<0 THEN RANDOMIZE TIMEDATE MOD 32767
21350   IF Mski THEN
21360     PRINT " ";Gens;"generations could expand to fill a cube sized";
21370     LET Agen=(2.0)^(Gens+1.0)          ! Pixel cube edge
```

```
21380     PRINT " ";VAL$(Agen);"x";VAL$(Agen);"x";VAL$(Agen);"."
21390   END IF
21400   IF Shv=0 THEN                        ! generate pattern fills
21410     FOR Gski=1 TO Gpcs                 ! up to 9 species
21420       FOR Sski=0 TO Qski               ! bit ordering of Sski is (Z,Y,X)B
21430         LET Ruls(Gski,Sski)=Gski       ! default
21440         DISP "Evolution from parent species [";VAL$(Gski);
21450         DISP "], sector (";VAL$(BIT(Sski,0));
21460         IF Qski>1 THEN DISP ",";VAL$(BIT(Sski,1));
21470         IF Qski>3 THEN DISP ",";VAL$(BIT(Sski,2));
21480         DISP ") is to species? (default=[";VAL$(Gski);
21490         INPUT "])",Ruls(Gski,Sski)
21500         LET Ruls(Gski,Sski)=INT(.5+Ruls(Gski,Sski))
21510       NEXT Sski
21520     NEXT Gski
21530   ELSE
21540     IF Rset=1 THEN                     ! get shuffle fills
21550       FOR Gski=1 TO Gpcs
21560         LET Mdl=0
21570         LET Kski=0
21580         FOR Sski=Clmx TO 1 STEP -1
21590           WHILE Mdl<1 AND Kski<=Gpcs
21600             LET Kski=Kski+1            ! current species fill
21610             IF Mski THEN
21620               DISP "How many descendants for species [";VAL$(Kski);
21630               DISP "] evolve from parent species [";VAL$(Gski);"]";
21640               DISP " (up to ";VAL$(Sski);
21650               INPUT ")",Mdl
21660             ELSE
21670               IF Kski=Gski THEN
21680                 LET Mdl=(1-Prpt)*Qski+.5
21690               ELSE
21700                 LET Mdl=Prpt*Qski
21710               END IF
21720             END IF                     ! endif, Mski=0 test
21730           END WHILE                    ! endif, Mdl<1 test
21740           LET Ruls(Gski,Clmx-Sski)=Kski
21750           LET Mdl=Mdl-1
21760         NEXT Sski
21770       NEXT Gski
21780     ELSE                               ! else, fill by demands
21790       LET Ruls(0,0)=0
21800       LET Nsft=0
21810       IF Mski THEN
21820         DISP "Establish all species with same probable descendancy";
21830         INPUT " pattern? 0=No 1=yes ",Nsft
21840       ELSE
21850         LET Nsft=1
21860       END IF
21870       LET Nsft=(Nsft>0)
21880       IF Nsft THEN
21890         IF Mski THEN
21900           DISP "What is fractional chance of parents descending";
21910           DISP " other species? (default=";VAL$(Prpt);
21920           INPUT ")",Prpt
```

```
21930           IF Prpt<0 OR Prpt>1 THEN Prpt=FRACT(Prpt)
21940         END IF                       ! endif, Mski else Prpt defaults
21950         IF Gpcs=1 THEN
21960           LET Ruls(1,1)=1
21970         ELSE
21980           FOR Gski=1 TO Gpcs
21990             FOR Sski=1 TO Gpcs
22000               IF Sski=Gski THEN
22010                 LET Ruls(Gski,Sski)=1-Prpt
22020               ELSE
22030                 LET Ruls(Gski,Sski)=Prpt/(Gpcs-1)
22040               END IF
22050             NEXT Sski
22060           NEXT Gski
22070         END IF
22080       END IF                         ! endif, Nsft test
22090       IF NOT Nsft THEN
22100         FOR Gski=1 TO Gpcs           ! up to 9 species
22110           LET Vski=0
22120           LET Ruls(Gski,0)=0
22130           FOR Sski=1 TO Gpcs         ! up to 9 species evolutions
22140             LET Ruls(Gski,Sski)=1/Gpcs! default
22150             IF Mski THEN
22160               DISP "Occupation most probable";
22170               DISP " for species [";VAL$(Sski);
22180               DISP "] descending from parent species [";VAL$(Gski);
22190               INPUT "]?",Ruls(Gski,Sski)
22200             ELSE
22210               IF Gski=Sski THEN
22220                 LET Ruls(Gski,Sski)=1-Prpt
22230               ELSE
22240                 LET Ruls(Gski,Sski)=Prpt/(1-Prpt)
22250               END IF
22260             END IF
22270             LET Vski=Vski+Ruls(Gski,Sski)
22280           NEXT Sski
22290           IF Vski<>0 THEN Vski=1/Vski
22300           IF Vski<>0 THEN            ! normalize evolution requests
22310             FOR Sski=1 TO Gski
22320               LET Ruls(Gski,Sski)=Ruls(Gski,Sski)*Vski
22330             NEXT Sski
22340           END IF
22350         NEXT Gski
22360       END IF                         ! endif, not Nsft test
22370     END IF                           ! endif, Rset test
22380   END IF                             ! endif, Shv=0 test
22390   IF Mski THEN                       ! imprint?
22400     DISP "Use which species as the STARTER evolutionary pattern?";
22410     DISP " (0=from user, default=";VAL$(Vlv);
22420     INPUT ")",Vlv
22430     LET Relay(4)=Vlv
22440   END IF
22450   IF Shv=0 THEN
22460     FOR Sski=0 TO Qski               ! set up of 2x2x2 Pixel fill
22470       LET Xski=BIT(Sski,0)+BIT(Sski,1)*Mlk1+BIT(Sski,2)*Mlk2+1
```

```
22480       IF Vlv=0 THEN
22490         LET Pixl(Xski)=BIT(BIT(Sski,0)+BIT(Sski,1)+BIT(Sski,2),0)+1
22500         DISP "Give STARTER species at Pixel(";VAL$(BIT(Sski,0)+1);
22510         IF Qski>1 THEN DISP ",";VAL$(BIT(Sski,1)+1);
22520         IF Qski>3 THEN DISP ",";VAL$(BIT(Sski,2)+1);
22530         INPUT ")",Pixl(Xski)
22540       ELSE
22550         LET Pixl(Xski)=Ruls(Vlv,Sski)
22560       END IF
22570     NEXT Sski
22580   ELSE                               ! initial random fill Shv<>0
22590     IF Rset THEN                     ! fill pixels directly fr Ruls
22600       LET Mdl=0
22610       LET Kski=0
22620       FOR Sski=Clmx TO 1 STEP -1
22630         WHILE Mdl<1 AND Kski<=Gpcs
22640           LET Kski=Kski+1            ! current species fill
22650           IF Mski THEN
22660             DISP "How many STARTER pixels for species [";VAL$(Kski);
22670             DISP "] (up to ";VAL$(Sski);
22680             INPUT ")",Mdl
22690           ELSE
22700             IF Kski=Gski THEN
22710               LET Mdl=(1-Prpt)*Qski+.5
22720             ELSE
22730               LET Mdl=Prpt*Qski
22740             END IF
22750           END IF                     ! endif, Mski=0 test
22760         END WHILE                    ! endif, Mdl<1 test
22770         LET Pixl(Sski)=Kski
22780         LET Mdl=Mdl-1
22790       NEXT Sski
22800     ELSE                             ! else Rset=0, get volume fractions
22810       IF Mski THEN
22820         LET Vski=0
22830         FOR Sski=1 TO Gpcs
22840           IF Vlv=0 THEN DISP "STARTER/";
22850           DISP "CONVERGENCE occupation value";
22860           DISP " saught for species [";VAL$(Sski);
22870           INPUT "]",Dmd(Sski)
22880           LET Vski=Vski+Dmd(Sski)
22890         NEXT Sski
22900         IF Vski=0 THEN
22910           LET Vski=1/Gpcs
22920         ELSE
22930           LET Vski=1/Vski
22940         END IF
22950         FOR Sski=1 TO Gpcs           ! normalize volume fractions
22960           LET Dmd(Sski)=Dmd(Sski)*Vski
22970         NEXT Sski
22980       END IF                         ! endif, else use default Dmd
22990       LET Dv=SUM(Dmd)
23000       IF Dv=0 THEN                   ! if 0,form default demands
23010         FOR Sski=1 TO Gpcs
23020           IF Vlv=0 OR Gpcs=1 THEN
```

```
23030            LET Dmd(Sski)=1/Gpcs     ! flat Demands
23040          ELSE
23050            IF Sski=Vlv THEN
23060              LET Dmd(Sski)=1-Prpt  ! proportionate Demands
23070            ELSE
23080              LET Dmd(Sski)=Prpt/(Gpcs-1)
23090            END IF
23100          END IF
23110        NEXT Sski
23120      END IF
23130      LET Vski=0
23140      LET Kski=0
23150      FOR Sski=1 TO Gpcs            !  fill pixels according to demands
23160        IF Vlv=0 THEN
23170          LET Vski=Vski+Dmd(Sski)*Clmx
23180        ELSE
23190          LET Vski=Vski+Ruls(Vlv,Sski)*Clmx
23200        END IF
23210        WHILE Kski<INT(Vski+.5)
23220          LET Kski=Kski+1
23230          LET Pixl(Kski)=Sski
23240        END WHILE
23250      NEXT Sski
23260    END IF                           ! endif, getting vol fractions
23270    FOR Sski=1 TO SHIFT(Clmx,1)
23280      REPEAT                         ! uses 1st 2 bit triples=6 bits total
23290        LET Nsft=INT(RND*64)
23300      UNTIL Nsft<64                  ! & mask out what is needed
23310      LET Mdl=BINAND(SHIFT(Nsft,3),Qski)+1
23320      LET Nsft=BINAND(Nsft,Qski)+1
23330      LET Kski=Pixl(Mdl)
23340      LET Pixl(Mdl)=Pixl(Nsft)
23350      LET Pixl(Nsft)=Kski
23360    NEXT Sski
23370    FOR Sski=Qski TO 0 STEP -1       ! move Pixels outward
23380      LET Xski=BIT(Sski,0)+BIT(Sski,1)*Mlk1+BIT(Sski,2)*Mlk2+1
23390      LET Pixl(Xski)=Pixl(Sski+1)
23400      IF Xski<>Sski+1 THEN LET Pixl(Sski+1)=0
23410    NEXT Sski
23420  END IF                             ! endif, Shv=0 test, imprinting
23430  IF Rset 1 THEN                     ! form numbering sequence
23440    FOR Sski=0 TO Qski               ! to be shuffled later
23450      LET Ncs(Sski)=Sski
23460    NEXT Sski
23470  END IF
23480  LET Dv=0                           ! Total servo volume request
23490  IF Shv<>0 AND NOT Rset THEN
23500    LET Dv=SUM(Dmd)
23510    IF Dv<>1 THEN PRINT " Norm? SERVO vol =";Dv;"in SUB evolv"
23520  END IF
23530  FOR Gski=1 TO Gens
23540    IF Dv>0 THEN
23550      MAT Pcs=(0)
23560      LET Kski=0
23570      FOR Sski=1 TO Cnk              ! tally for adjusting
```

```
23580         LET Pcs(Pixl(Sski))=Pcs(Pixl(Sski))+1
23590         IF Pixl(Sski)>0 THEN Kski=Kski+1
23600       NEXT Sski
23610       FOR Sski=1 TO Gpcs
23620         IF Kski=0 THEN
23630           LET Ruls(0,Sski)=0
23640         ELSE
23650           LET Ruls(0,Sski)=0          ! estimate probable fills
23660           FOR Xski=1 TO Gpcs
23670             LET Ruls(0,Sski)=Ruls(0,Sski)+(Pcs(Xski)/Kski)*Ruls(Xski,Sski)
23680           NEXT Xski                   ! servo coefficients
23690           LET Ruls(0,Sski)=(Dmd(Sski)-Ruls(0,Sski))
23700         END IF
23710       NEXT Sski
23720     END IF
23730     FOR Zski=BINAND(-2,Zmax+1)-1 TO 1 STEP -2 ! 2 fold
23740       FOR Yski=BINAND(-2,Ymax+1)-1 TO 1 STEP -2
23750         FOR Xski=BINAND(-2,Xmax+1)-1 TO 1 STEP -2
23760           LET Fadr=Xski+(Yski-1)*Xmax+(Zski-1)*Mlk2 ! new address
23770           LET Sadr=SHIFT(Xski+1,1)+SHIFT(Yski-1,1)*Xmax
23780           LET Sadr=Sadr+SHIFT(Zski-1,1)*Mlk2 ! source address
23790           LET Rlr=Pixl(Sadr)         ! Pixel to be 2 folded
23800           LET Pixl(Sadr)=0
23810           IF Shv<>0 AND Rset=1 THEN ! shuffle the sequencing
23820             FOR Sski=1 TO SHIFT(Clmx,1)
23830               REPEAT                 ! uses 1st 2 bit triples=6 bits total
23840                 LET Nsft=INT(RND*64)
23850               UNTIL Nsft<64          ! & mask out what is needed
23860               LET Mdl=BINAND(SHIFT(Nsft,3),Qski)
23870               LET Nsft=BINAND(Nsft,Qski)
23880               LET Kski=Ncs(Mdl)
23890               LET Ncs(Mdl)=Ncs(Nsft)
23900               LET Ncs(Nsft)=Kski
23910             NEXT Sski
23920           END IF
23930           FOR Sski=0 TO Qski
23940             LET Kski=BIT(Sski,0)+BIT(Sski,1)*Mlk1+BIT(Sski,2)*Mlk2
23950             LET Kski=Kski+Fadr       ! Final new address
23960             IF Rlr=0 OR Rlr>Gpcs THEN
23970               LET Pixl(Kski)=0
23980             ELSE
23990               IF Shv=0 OR Rset THEN ! Fill by numbers in Ruls
24000                 LET Pixl(Kski)=Ruls(Rlr,Ncs(Sski))
24010               ELSE                   ! Fill by probabilities
24020                 LET Agen=RND
24030                 LET Vski=0
24040                 LET Mdl=0
24050                 REPEAT
24060                   LET Mdl=Mdl+1
24070                   LET Vski=Vski+Ruls(Rlr,Mdl)+Ruls(0,Mdl)
24080                 UNTIL Vski>=Agen OR Mdl=Gpcs
24090                 LET Pixl(Kski)=Mdl
24100               END IF
24110             END IF
24120           NEXT Sski
```

```
24130         NEXT Xski
24140       NEXT Yski
24150     NEXT Zski
24160   NEXT Gski
24170   LET Kski=0
24180   FOR Sski=1 TO Cnk
24190     LET Kski=Kski+(Pixl(Sski)=0)
24200   NEXT Sski
24210   IF Kski>0 THEN PRINT Kski;"empty? pixels returned, SUB evolv"
24220   DEALLOCATE Ruls(*)                 ! Bye-bye variable sub arrays
24230 SUBEND
```

ELECTRONICS TECHNOLOGY AND DEVICES LABORATORY
MANDATORY DISTRIBUTION LIST
CONTRACT OR IN-HOUSE TECHNICAL REPORTS

Defense Technical Information Center*
ATTN: DTIC-FDAC
Cameron Station (Bldg 5)
Alexandria, VA 22304-6145

(*Note: Two copies for DTIC will be sent from STINFO office.)

Director
US Army Material Systems Analysis Actv
ATTN: DRXSY-MP
001 Aberdeen Proving Ground, MD 21005

Commander, AMC
ATTN: AMCDE-SC
5001 Eisenhower Ave.
001 Alexandria, VA 22333-0001

Commander, LABCOM
ATTN: AMSLC-CG, CD, CS (in turn)
2800 Powder Mill Road
001 Adelphi, MD 20783-1145

Commander, LABCOM
ATTN: AMSLC-CT
2800 Powder Mill Road
001 Adelphi, MD 20783-1145

Commander,
US Army Laboratory Command
Fort Monmouth, NJ 07703-5601
1 - SLCET-DD
1 - SLCET-DT (M. Howard)
1 - SLCET-DR-B
22 - Originating Office

Commander, CECOM
R&D Technical Library
Fort Monmouth, NJ 07703-5703
1 - ASQNC-ELC-IS-L-R (Tech Library)
3 - ASQNC-ELC-IS-L-R (STINFO)

Advisory Group on Electron Devices
ATTN: Documents
2011 Crystal Drive, Suite 307
002 Arlington, VA 22202

Director
Naval Research Laboratory
ATTN: CODE 2627
001 Washington, DC 20375-5000

Cdr, PM JTFUSION
ATTN: JTF
1500 Planning Research Dr
001 McLean, VA 22102

Rome Air Development Center
ATTN: Documents Library (TILD)
001 Griffis AFB, NY 13441

Deputy for Science & Technology
Office, Asst Sec Army (R&D)
001 Washington, DC 20310

HQDA (DAMA-ARZ-D/Dr. F.D. Verderame)
001 Washington, DC 20310

Dir, Electronic Warfare/Reconnaissance
Surveillance & Target Acquisition Dir
ATTN: AMSEL-RD-EW-D
001 Fort Monmouth, NJ 07703-5206

Dir, Reconnaissance Surveillance &
Target Acquisition Systems Dir
ATTN: AMSEL-RD-EW-DR
001 Fort Monmouth, NJ 07703-5206

Cdr, Marine Corps Liaison Office
ATTN: AMSEL-LN-MC
001 Fort Monmouth, NJ 07703-5033

Dir, US Army Signals Warfare Dir
ATTN: AMSEL-RD-SW-OS
Vint Hill Farms Station
001 Warrenton, VA 22186-5100

Dir, Night Vision & Electro-Optics Dir
CECOM
ATTN: AMSEL-RD-NV-D
001 Fort Belvoir, VA 22060-5677

Cdr, Atmospheric Sciences Lab
LABCOM
ATTN: SLCAS-SY-S
001 White Sands Missile Range, NM 88002

Cdr, Harry Diamond Laboratories
ATTN: SLCHD-CO, TD (in turn)
2800 Powder Mill Road
001 Adelphi, MD 20783-1145